Wonderware® FactorySuite™
InTouch® 参考指南

用户指南

修订版

2001 年 6 月

Wonderware Corporation

100 Technology Drive
Irvine, CA 92618
U.S.A.
(949) 727-3200
http://www.wonderware.com

商标

目录

简介

本参考指南按字母顺序编排，向您介绍 InTouch 产品套件中每个标记名点域 (**.field**)、窗口控件属性、报警对象属性、系统标记名和 QuickScript 函数的详细信息。其中包括配方管理器、SPC Pro 和 SQL Access Manager 等 InTouch 附加功能。本参考指南由以下章节组成：

- 第 1 章“系统标记名”，详细介绍所有 InTouch 预定义系统标记名。

注意：所有系统标记名以美元符号 ($) 开头。

- 第 2 章“点域”，详细介绍标记名、报警、分布式报警、窗口控件、系统、历史趋势及其它对象的标记名**点域**。
- 第 3 章“InTouch QuickScript 函数”，详细介绍内置 InTouch QuickScript 函数（字符串、数学、系统、历史、窗口控件、分布式报警及其它函数）。
- 第 4 章“OLE Automation 与 InTouch”，简要概括 OLE Automation 的结构。

文档约定

约定实例	描述
.ChartStart	一般情况下，粗体文本表示点域、系统标记名或函数。
Tagname	在语法中，斜体字母表示您应提供信息的占位符。
{ **.HiLimit** | **.HiHiLimit** }	在语法中，大括号和竖杠表示从两个或多个项目中选择。
[ErrorMessage=]	方括号中的项目是可选项。

为方便参考，每个系统标记、属性、点域和脚本函数标题的右边列出了使用分类（安全性、报警、应用程序等）。例如：

$AccessLevel 安全性

关于本手册

如果您阅读的是本手册的联机版，则可以在看到绿色文本时，单击该文本来跳到相关的章节。如果您要在跳转后回到原来的章节，可以使用所提供的“后退”选项。

这些“提示”可以告诉您如何用更简便快捷的方法来完成某项功能或任务。

*《InTouch 用户指南》*将帮助您熟悉 WindowMaker 开发环境及其工具，请参阅第 1 章“WindowMaker 程序元素”。要了解如何使用窗口、图形对象、向导和 ActiveX 控件等元素，请阅读第 2 章“使用 WindowMaker”。有关使用 InTouch QuickScript 的详细信息，请参阅第 6 章“在 InTouch 中创建 QuickScript”。

有关运行时环境 (WindowViewer) 的详细说明，请参阅*《InTouch 运行时用户指南》*。

有关附加程序 SPC Pro 的详细信息，请参阅您的*《SPC Pro 用户指南》*。

有关附加程序“配方管理器”的详细信息，请参阅您的*《配方管理器用户指南》*。

有关附加程序 SQL Access Manager 的详细信息，请参阅您的*《SQL Access Manager 用户指南》*。

*《FactorySuite 系统管理员指南》*也为您提供了有关 FactorySuite 中的一般组件、系统要求、网络连接注意事项、产品集成、技术支持等方面的完整信息。

FactorySuite 软件包还提供了所有 FactorySuite 组件的联机版手册。

注意：要查看或打印联机手册，您必须安装 Adobe Acrobat Reader（4.0 或以上版本）。

假设

本手册假设您：

- 已经熟悉 Windows 2000 和（或）Windows NT 操作系统工作环境。
- 懂得如何使用鼠标、Windows 菜单、选择选项和访问联机帮助。
- 有使用编程语言和宏语言的经验。最好理解一些编程概念，如变量、语句、函数和方法等。

技术支持

Wonderware 技术支持提供许多支持选项，来回答有关 Wonderware 产品及其实施的问题。

在联系技术支持前，请先参考*《InTouch 用户指南》*中的有关章节，以寻求您使用 InTouch 系统时所遇问题的可能解决方法。如果您觉得有必要求助于技术支持，请提供以下信息：

1. 您的软件序列号。
2. 您所运行的 InTouch 版本。
3. 您所使用的操作系统类型和版本。例如 Microsoft Windows NT 4.0 SP4 workstation。
4. 描述所遇系统错误消息的准确用词。
5. 来自 Wonderware Logger、Microsoft Diagnostic utility (MSD) 或任何其它诊断应用程序的任何相关输出列表。
6. 您所尝试的解决方法的细节和结果。
7. 有关如何重现问题的细节。
8. 如果所遇问题是老问题，请提供指定的 Wonderware 技术支持案例号。

有关技术支持的详细信息，请参阅您的联机*《FactorySuite 系统管理员指南》*。

您的 FactorySuite 许可证

要查看您的 FactorySuite 系统许可证信息，可以从 WindowMaker 帮助菜单的“关于”对话框中启动许可证查看程序进行查看。

➢ 要打开许可证实用程序：

1. 在 WindowMaker 的**“帮助”**菜单上，选择**“关于”**命令。
2. 单击**“View License”**（查看许可证）。**“License Utility - LicView”**（许可证实用程序 - LicView）对话框出现。

有关许可证查看实用程序的详细信息，请参阅您的*《FactorySuite 系统管理员指南》*。

第 1 章

系统标记名

InTouch 提供了若干个预定义的系统标记名。这些标记名内置于 InTouch 中并在前面加上一个美元 ($) 符号加以识别。在运行期间，InTouch 作用于这些标记名值来回应系统事件。任何可使用 InTouch 标记名的地方都可以使用系统标记名，如 QuickScript 的动画链接中。

$AccessLevel 安全性

定义当前登录用户的访问级别。

用法 **$AccessLevel**

备注 只读标记名，其值取决于指定给 InTouch 当前登录用户安全特性配置文件的访问级别。此特性配置文件可以通过选择 WindowViewer 上的“配置用户”菜单进行访问。

$AccessLevel 的实际数值对于 WindowViewer 没有任何意义。所需的任何“安全性”必须应用程序设计者建立。通过使用 **$AccessLevel**，可以进行多项设置以启用系统内的安全性。

数据类型 整型（只读）

有效值 0 到 9999

实例 下面的语句用于可见性链接，以使某个对象（如按钮），可以根据登录用户的访问级别进行显示或隐藏：

```
$AccessLevel >= 2000;
{通过基于 AccessLevel 的表达式，对象可以与一个“失效”链接关联。}
$AccessLevel < 5411;

IF $AccessLevel <=500 THEN
    Show "拒绝访问"; {弹出拒绝访问窗口}

ELSE
    Show "拒绝访问"; {弹出允许访问窗口}

ENDIF;
```

参见 **$Operator, $OperatorEntered, $PasswordEntered; $ConfigureUsers**

$AlarmLogging 报警

在运行期间重新初始化报警记录和打印。此标记名等同于 WindowViewer“**特别**”菜单上的“**重新启动报警日志**”命令。

用法
```
$AlarmLogging = 1;
```

备注

将此值设为 1 可在运行时间重新初始化报警记录和打印。

注意：$AlarmLogging 标记名不能用于关闭报警记录或打印。它被严格用于替换 WindowViewer“**特殊**”菜单中的“**重新启动报警日志**”。**$AlarmLogging** 值总是显示为 0（关闭），即使正在进行记录。设置 **$AlarmLogging** 为不等于 1 的值是无意义的，其结果没有定义。

数据类型

离散型（只写）

有效值

1

实例

下面的语句所执行的动作与选择 WindowViewer“**特别**”菜单上的“**重新启动报警日志**”相同。

```
$AlarmLogging = 1;
```

参见

$HistoricalLogging

$AlarmPrinterError 报警

报警打印机错误。

用法
```
$AlarmPrinterError
```

备注

如报警打印机出现错误，则值为 1。

数据类型

离散型（只读）

有效值

0 = 报警打印机无错误
1 = 报警打印机出错

实例

下面的语句如用作“模拟量显示链接”中的表达式，则会在报警打印机出错时显示 1；否则显示 0。

```
IF $AlarmPrinterError == 1 THEN
   DisplayMessageTag = "报警打印机出错";
ELSE
   DisplayMessageTag = "报警打印机出错";
ENDIF;
```

参见

$AlarmPrinterNoPaper, $AlarmPrinterOffline, $AlarmPrinterOverflow

$AlarmPrinterNoPaper 报警

报警打印机缺纸。

用法 `$AlarmPrinterNoPaper`

备注 如报警打印机缺纸，则值为 1。

数据类型 离散型（只读）

有效值 0 = 报警打印机有纸
1 = 报警打印机缺纸

实例 本语句如用作模拟量显示链接中的表达式，在报警打印机缺纸时显示 1；否则，它显示 0。

```
$AlarmPrinterNoPaper
```

在“**为真时**”条件动作脚本中使用：

条件：

```
$AlarmPrinterOffline == 1
```

脚本：

```
show "window";
playsound("c:\winnt\system32\alarm.wav",1);
```

参见 **$AlarmPrinterOffline, $AlarmPrinterError, $AlarmPrinterOverflow**

$AlarmPrinterOffline 报警

报警打印机脱机。

用法 `$AlarmPrinterOffline`

备注 如报警打印机脱机，则值为 1。

数据类型 离散型（只读）

有效值 0 = 报警打印机联机
1 = 报警打印机脱机

实例 本语句如用作模拟量显示链接中的表达式，在报警打印机脱机时显示 1；否则，它显示 0。

```
$AlarmPrinterOffline
IF $AlarmPrinterOffline == 1 THEN
   Call PLCHorn( );
```

{打开车间的声音报警的 **Quickfunction**}

```
ENDIF;
```

参见 **$AlarmPrinterNoPaper, $AlarmPrinterError, $AlarmPrinterOverflow**

$AlarmPrinterOverflow 报警

报警打印机缓冲区溢出。

用法 `$AlarmPrinterOverflow`

备注 如果报警打印机出错（送至打印机的字符过多），则值为 1。

数据类型 离散型（只读）

有效值 0 = 报警打印机缓冲区未满
1 = 报警打印机缓冲区已满并溢出

实例 下面的语句如用作“模拟量显示链接”中的表达式，则会在报警打印机缓冲区已满时显示 1；否则显示 0。

`$AlarmPrinterOverflow`

参见 **$AlarmPrinterNoPaper, $AlarmPrinterOffline, $AlarmPrinterError**

$ApplicationChanged 应用程序

指示网络应用程序开发 (NAD) 结构中的主应用程序已改变。

用法 `$ApplicationChanged`

备注 每当通过选择 WindowViewer **“特别”**菜单上的**“通知客户”**而产生更新信号时，此标记名将变为 1。在应用程序更新后，重置为 0；用于分布式应用程序维护。

数据类型 离散型（只读）

实例 下面的语句如用在数据改变脚本的标记名点域中，会导致 QuickScript 主体执行。QuickScript 主体可能显示一个窗口，通知用户重新启动 WindowViewer，以使更改生效。

`$ApplicationChanged`

参见 **$ApplicationVersion**

$ApplicationVersion 应用程序

包含应用程序的当前版本号。每当标记名或 QuickScript 改变、添加或删除时，此数值都会改变。

用法 **$ApplicationVersion**

备注 无

数据类型 实型（只读）

实例 用于“模拟量显示链接”时，此系统标记将显示在 WindowViewer 中运行的应用程序的当前版本。

$ApplicationVersion

参见 **$ApplicationChanged**

$ChangePassword 安全性

显示“**改变口令**”对话框。相当于选择 WindowViewer“**特别**”菜单上的“**安全性**”然后再选择“**改变口令**”命令。

用法 **$ChangePassword**

备注 设置值为 1 可显示“**改变口令**”对话框。一旦关闭对话框，InTouch 会将值自动重置为 0。设置此系统标记名为 1 以外的值没有意义，其结果未定义。

数据类型 离散型（只写）

有效值 1

实例 创建一个离散按钮，以允许用户显示“**改变口令**”对话框。按钮应添加一个“离散按钮”链接，并选定“**置位**”选项。当按下按钮时，**$ChangePassword** 置为 1，同时显示“**改变口令**”对话框。

参见 **$AccessLevel, $OperatorEntered, $PasswordEntered, $Operator, $ConfigureUsers**

$ConfigureUsers 安全性

显示通用的“**配置用户**”对话框。相当于选择 WindowViewer“**特别**”菜单上的“**安全性**”然后再选择“**改变口令**”命令。

用法 `$ConfigureUsers`

备注 设置值为 1 可显示“**配置用户**”对话框。一旦关闭对话框，InTouch 会将值自动重置为 0。要显示此对话框，用户的 **$AccessLevel** 必须 >9000。设置此系统标记名为 1 以外的值没有意义，其结果未定义。

数据类型 离散型（只写）

有效值 1

实例 创建一个离散按钮，以允许用户显示“**改变口令**”对话框。按钮应添加一个“离散按钮”链接，并选定“**置位**”选项。当按下按钮时，**$ConfigureUsers** 置为 1，同时显示“**配置用户**”对话框。

参见 **$Operator, $OperatorEntered, $ChangePassword, $PasswordEntered, $AccessLevel**

$Date 系统

包含自 1970 年 1 月 1 日以来的天数（整数）。

用法 `$Date`

备注 无

数据类型 整型（只读）

实例

```
StringFromTime(($Date*86400)+($Time/1000),3);
```

{返回当前时间}

参见 **$DateTime, $DateString, $Day, $Month, $Year**

$DateString 系统

包含以 win.ini 文件中设置的格式表示的日期。

用法
```
$DateString
```

备注
日期格式通过 Windows 控制面板或双击 Windows 任务栏上的“时钟”进行设置。

数据类型
内存消息（只读）

实例
```
If StringRight($DateString,2)== "00" THEN
   LogMessage("The Year 2000!");
ENDIF;
```

参见
$Date, $DateTime, $Day, $Month, $Year, $Time, $TimeString

$DateTime 系统

包含自 1970 年 1 月 1 日以来的天数（小数）。

用法
```
$DateTime
```

备注
无

数据类型
实型（只读）

实例
```
dayvalue = 86400 * $DateTime;

If StringFromTime(Dayvalue,4) == "Fri" THEN
   DisplayMessageTag = "It's Friday!";
ENDIF;
```

参见
$Date, $DateString, $Day, $Month, $Year, $Time, $TimeString

$Day 系统

包含当前月的当前日。

用法
```
$Day
```

备注
包含介于 1 和 31 之间的值。

数据类型
整型（只读）

实例
```
If $Day == 15 THEN
   Show "Mid-Month Washdown Window";
ENDIF;
```

参见
$DateTime, $DateString, $Date, $Month, $Year

$HistoricalLogging 历史

监控历史记录的启动和终止。相当于选择 WindowViewer“**特别**”菜单上的“**重新启动历史记录**”或“**停止历史记录**”命令。

用法 `$HistoricalLogging`

备注 将此系统标记名设置为 0 将停止历史记录。设置此值为 1 可重新启动历史记录。除非在“**历史记录属性**”对话框中选定了“**允许历史记录**”选项，否则您不能通过 **$HistoricalLogging** 启动历史记录。

数据类型 离散型（读/写）

实例

```
IF (InfoDisk("C",4,$Second)/1024)<50 THEN
{当磁盘空间小于 50MB 时停止历史记录}
$HistoricalLogging = 0;
ENDIF;
```

参见 **$AlarmLogging**

$Hour 系统

包含当天的小时值。

用法 `$Hour`

备注 包含介于 0 和 23 之间的值。

数据类型 整型（只读）

实例 为真时条件脚本：

```
$Hour == 20 AND $Second == 30
PrintWindow("Day Batch Summary",1,1,0,0,0);
```

参见 **$DateTime, $Minute, $Msec, $Second, $Time, $TimeString**

$InactivityTimeout 安全性

表示自动注销用户的配置时间已过。

用法 `$InactivityTimeout`

备注 当自动注销时间已过时，值为 1。要配置标记名时间，在“**特别**”菜单上，选择“**配置**”，然后单击“**WindowViewer**”或者在应用程序浏览器的“**配置**”下，双击“**WindowViewer**”。“**WindowViewer 属性**”对话框出现并显示“**通用**”属性页。在“闲置”组中，输入以秒为单位的时间值。

注意：对于 Active-X 控件、OLE 自动化控件以及 SPC 向导，闲置定时器不会重置。

数据类型 离散型（只读）

参见 **$InactivityWarning**

实例 为真时条件脚本：

条件：
```
$InactivityTimeout == 1
```

脚本：
```
Show "You have been logged off window";
```

参见 **$InactivityWarning**

$InactivityWarning 安全性

表示警告用户自动注销用户的配置时间已过。

用法 **$InactivityWarning**

备注 当注销事件将要发生时，值为 1。闲置计时器只能通过单击鼠标或键盘活动来重置。要配置标记名时间，在“**特别**”菜单上，选择“**配置**”，然后单击“**WindowViewer**”或者在应用程序浏览器的“**配置**”下，双击“**WindowViewer**”。“**WindowViewer 属性**”对话框出现并显示“**通用**”属性页。在“闲置”组中，输入以秒为单位的时间值。

注意：对于 Active-X 控件、OLE 自动化控件以及 SPC 向导，闲置定时器不会重置。

数据类型 离散型（只读）

实例 为真时条件脚本：

```
If $InactivityWarning == 1 THEN
   Show "You are about to be logged off-window";
ENDIF;
```

参见 **$InactivityTimeOut**

$LogicRunning 系统

监控脚本的运行。相当于选择 WindowViewer“**逻辑**”菜单上的“**开始逻辑**”或“**停止逻辑**”命令。

注意：您不能停止当前执行的异步 QuickFunction。但是，可以防止执行新的 QuickFunction。

用法 **$LogicRunning**

备注 设置值为 1 以启动 QuickScript 逻辑。设置值为 0 以停止 QuickScript 逻辑。

数据类型 离散型（读/写）

$Minute 系统

包含当前小时的分钟值。

用法 **$Minute**

备注 包含介于 0 和 59 之间的值。

数据类型 整型（只读）

实例 在动画链接值输出字符串表达式中：

```
IF InfoFile("C:\InTouch.32\WIZ.INI",1,$Minute)==1 THEN
   LogMessage("The File Exists!");
ENDIF;
```

参见 **$DateTime, $Hour, $Msec, $Second, $Time, $TimeString**

$Month 系统

包含当前月份值。

用法 **$Month**

备注 包含介于 1 和 12 之间的值。

数据类型 整型（只读）

实例

```
IF $Month==10 THEN
   CurrentMonthName = "October";
ENDIF;
```

参见 **$Date, $Day, $Year**

$Msec 系统

包含当前的毫秒值。改变“计时间隔”和“更新时间变量”的设置，可增大此系统标记名的更新速率。在“**特别**”菜单上，指向“**配置**”，然后单击“**WindowViewer**”，或者在应用程序浏览器的“**配置**”下，双击“**WindowViewer**”。“**WindowViewer 属性**”对话框出现并显示“**通用**”属性页。

用法 `$Msec`

备注 包含介于 0 和 999 之间的值。

数据类型 整型（只读）

参见 **$DateTime, $Minute, $Second, $Hour, $Time, $TimeString**

$NewAlarm 报警

指示发生新报警。

用法 `$NewAlarm`

备注 当发生新报警时，值为 1。此值仅在发生新的本地报警时设置，不能用于远程报警。

数据类型 离散型（读/写）

实例 此标记名可链接到 **PlaySound** 逻辑函数，以产生声音报警。可创建一个确认按钮，以允许操作员将此值重置为 0，并确认报警。

$ObjHor 系统

包含所选对象中心的水平象素位置。

用法 `$ObjHor`

备注 无

数据类型 整型（只读）

参见 **$ObjVer**

$ObjVer　系统

包含所选对象中心的垂直象素位置。

用法　`$ObjVer`

备注　无

数据类型　整型（只读）

参见　**$ObjHor**

$Operator　安全性

控制用户执行指定功能的能力。

用法　`$Operator`

备注　包含当前登录的用户名。

数据类型　消息型（只读）

实例　通过输入下列脚本，可控制对特定窗口的访问：

```
IF $Operator == "DayShift" THEN
   Show "Control Panel Window";
ELSE
   Show "Wrong Operator";
ENDIF;
```

参见　**$OperatorEntered, $AccessLevel, $PasswordEntered, $ChangePassword, $ConfigureUsers**

$OperatorEntered　安全性

用于输入有效用户名。

用法　`$OperatorEntered`

备注　创建自定义登录窗口。此标记名可链接触控输入对象和（或）QuickScript，以便设置用户的“用户名”。

注意：当 **$OperatorEntered** 有效时，**$AccessLevel** 和 **$Operator** 将设置为其预定义值。

数据类型　消息型（读/写）

参见　**$AccessLevel, $Operator, $PasswordEntered, $ChangePassword, $ConfigureUsers**

$PasswordEntered 安全性

用于输入有效口令。

用法 `$PasswordEntered`

备注 此系统标记名总是返回一个空的字符串。与 **$PasswordEntered** 有关的显示链接总是显示为空白。因为此标记名总是返回空字符串，所以 $PasswordEntered 无法触发数据更改脚本。可用于创建一个自定义登录窗口。此标记名可链接触控输入对象和（或）QuickScript，以便设置用户的“口令”。

注意：当 **$OperatorEntered** 有效时，**$AccessLevel** 和 **$Operator** 将设置为其预定义值。

数据类型 消息型（只写）

参见 **$AccessLevel, $Operator, $OperatorEntered, $ChangePassword, $ConfigureUsers**

$Second 系统

包含当前秒值。

用法 `$Second`

备注 包含介于 0 和 59 之间的值。

数据类型 整型（只读）

实例 `Wave=100*Sin(6*$Second);`

参见 **$DateTime, $Minute, $Msec, $Hour, $Time**

$StartDdeConversations 系统

在运行期间，当“**特别**”菜单被禁用时，启动未初始化的对话。相当于选择 WindowViewer“**特别**”菜单上的“**启动未初始化的对话**”。

用法 `$StartDdeConversations`

备注 设置值 1 以启动未初始化的 DDE 对话。

数据类型 离散型（读/写）

$System 报警

缺省报警组。

用法 `$System`

备注 如果标记名未指定给特定报警组名，缺省条件下它将自动指定给此根报警组。所有定义的报警组都是 **$System** 的后代。

数据类型 系统报警组（只读）

实例 **$System.Ack = 1;{确认所有报警}**

$Time 系统

包含自午夜以来的以毫秒为单位的时间。

用法 `$Time`

备注 无

数据类型 整型（只读）

实例 **Sec_Midnight = $Time/1000;{自午夜以来的秒数}**

参见 **$DateTime, $Minute, $Msec, $Second, $Hour, $TimeString**

$TimeString 系统

包含以 WIN.INI 文件所设置的格式表示的时间。

用法 `$TimeString`

备注 时间格式通过 Windows 控制面板或双击 Windows 任务栏上的“时钟”进行设置。

数据类型 字符串（只读）

实例 `BatchStartString = $TimeString;`

参见 **$DateTime, $Minute, $Msec, $Second, $Hour, $Time**

$Year 系统

包含四位数的当前年份。

用法 `$Year`

备注 包含以下格式的年份：

1990

数据类型 整型（只读）

实例

```
CurrentYear = $Year;
NoYrsTill2000 = 2000 - CurrentYear;
```

参见 **$Day, $Month**

第 2 章

点域

InTouch 使用点域来显示诸如标记名、历史趋势和窗口控件等对象的属性。您可以使用这些点域来监视和修改上述属性。根据对象的不同，这些属性可通过以下两种方法中之一进行访问：直接通过 *Tagname.field* 语句，或间接地通过脚本函数。

除了窗口控制和分布式报警对象外的所有对象都使用 *Tagname.Field* 句法。要访问这些属性，您只需输入对象名称，后跟您要在 InTouch QuickScript 或动画链接中访问的属性。例如，要允许运行时改变标记名 **Analog_Tag** 的 HiHi 报警限，您可以创建一个“**模拟 – 用户输入**”触动链接，并将其应用到使用表达式 **Analog_Tag.HiHiLimit** 的按钮。在运行时，操作员只需单击按钮，即可为 **Analog_Tagname** 的 HiHi 报警限输入一个新值。

窗口控件和分布式报警对象使用 *GetPropertyX* 和 *SetPropertyX* 脚本函数，其中 X 是属性的数据类型（D=离散型，I=整数，M=消息型）。您也可以在 InTouch QuickScript 或动画链接中使用这些函数来访问属性。

有关 *GetProperty* 和 *SetProperty* QuickScript 函数的详细信息，请参阅第 3 章“InTouch QuickScript 函数”。

标记名类型

在 InTouch 标记名字典中定义标记名时，您必须根据其用法为每一个标记名指定一种类型。例如，如果一个标记名要读或写来自象 I/O 服务器这样的另一 Windows 应用程序的数值，它就必须是一个 I/O 类型的标记名。下面描述每种 InTouch 标记名类型及其用法。

内存类型标记名

内存标记名类型存在于您的 InTouch 应用程序内。您可以使用它们来生成系统常数和模拟，也可以用它们建立由其它 Windows 程序访问的计算变量。例如：您可以用初始值 3.1416 来定义内存标记名，或者您可以把配方储存在一组内存标记名内。在模拟方面，您能用内存标记名来控制背景脚本的动作。例如，您可以定义一个在某一动作脚本中被改变的内存标记名“COUNT”来为某一过程的当前的 STEP 生成各种动画效果。共有四种内存类型：

内存离散型

内部离散型标记名，值为 0（假，关）或 1（真，开）。

内存整型

值为 -2,147,483,648 到 2,147,483,647 之间的 32 位有符号整数。

内存实型

浮点（小数）内存标记名。浮点值可以介于 $\pm 3.4e^{38}$ 之间。所有浮点计算都按 64 位精度操作，但用 32 位来保存结果。

内存消息型

最长为 131 个字符的文本字符串标记名。

I/O 型标记名

所有从另一 Windows 程序读取其值或将其值写入另一 Windows 程序的标记名为 I/O 类型标记名。这包括所有来自可编程控制器、过程计算机的输入输出以及来自网络节点的数据。I/O 标记名可通过 Microsoft 动态数据交换(DDE)协议或 SuiteLink 传输协议来访问。

当一个读/写 I/O 类型标记名改变时，它将被立即写入远程应用程序。每当链接到远程应用程序的标记名的项改变时，标记名也可以从远程应用程序上更新。缺省时,所有的 I/O 标记名设置到读/写。然而，您可以通过选择“**标记名字典**”对话框中的“只读”选项将其限制为只读。共有四种 I/O 类型：

I/O 离散型

离散型输入/输出标记名，值为 0（假，关）或 1（真，开）。

I/O 整型

值为 -2,147,483,648 到 2,147,483,647 之间的 32 位有符号整数。

I/O 实型

浮点（小数）标记名。浮点值可以介于 $\pm 3.4e^{38}$ 之间。所有浮点计算都按 64 位精度操作，但用 32 位来保存结果。

I/O 消息型

最长为 131 个字符的文本字符串输入/输出标记名。

间接离散、间接模拟、间接消息

间接类型的标记名允许您创建一个窗口，并将该窗口内的标记名重新指定给多个源标记名。例如，您可以为一个窗口内基于一个改变值的所有标记创建一个“数据更改脚本”。

其它类型标记名

您还可以赋予标记名一些特殊标记名类型来执行复合函数，如动态报警显示、历史趋势、监控每一历史趋势笔所绘制的标记名。您还可以用间接标记名类型来将一个标记名赋予多个资源。这些特殊标记名类型描述如下。

Hist Trend

当生成一个历史趋势时，InTouch 需要用到 **Hist Trend** 类型的标记名。所有的与历史趋势关联的**点域**都可以运用到 **Hist Trend** 标记名中。

Tag ID

这是专用于历史趋势对象的特殊类型标记名。您可以用 Tag ID 类型的标记名来检索在历史趋势上所绘制的标记名信息。多数情况下，您会用 **Tag ID** 标记名来显示赋予指定笔的标记名的名称或，改变赋予该笔的标记名。

有关历史趋势的详细信息，请参阅您的联机《*InTouch 用户指南*》。

SuperTag

InTouch SuperTag 允许您定义复合标记名类型。您可最多用 64 个成员和两个嵌套级别来定义 SuperTag。成员的性能与通常的标记名完全相同。它们支持趋势、报警和所有标记名**点域**。

有关 SuperTag 的详细信息，请参阅您的《*InTouch 用户指南*》。

标记名类型与点域使用表

下表阐述了标记名类型与点域之间的快速引用关系。下面的脚注适用于下页的表格。

如果这些内存类型标记名是作为 I/O 点从*客户端*发出请求，则视图（服务器）会提供其本地时间标签，即请求值发送到客户端的时间值。然而，在本机看来内存标记名始终具有好的质量，并且没有时间标签，也就是说时间标签不会根据脚本或链接中的内存值改变而更新。

*在 InTouch 7.0 或以上版本中支持。

** 在 InTouch 7.1 或以上版本中支持。

*** 在 InTouch 7.11 或以上版本中支持。

标记名类型	内存型			I/O			间接		其它			
	离散型	整型	实型	离散型	整型	实型	离散型	模拟	报警组	Hist Trend	分布式报警对象/控件	Tag ID
点域名												
.Ack	X	X	X	X	X	X	X	X	X			
.AckDev***		X	X		X	X		X	X			
.AckDsc	X			X			X		X			
.AckROC***		X	X		X	X		X	X			
.AckValue***		X	X		X	X		X	X			
.Alarm	X	X	X	X	X	X	X	X	X			
.AlarmAccess **											X	
.AlarmAckModel***	X	X	X	X	X	X	X	X	X			
.AlarmClass **											X	
.AlarmComment ** Usage 1	X	X	X	X	X	X	X	X	X			X
.AlarmComment ** Usage 2											X	
.AlarmDate**											X	
.AlarmDev		X	X		X	X		X	X			
.AlarmDevCount		X	X		X	X		X	X			
.AlarmDevDeadband		X	X		X	X		X				
.AlarmDevUnAckCount		X	X		X	X		X	X			
.AlarmDisabled***	X	X	X	X	X	X	X	X	X			
.AlarmDsc	X			X			X		X			
.AlarmDscCount	X			X			X		X			
.AlarmDscDisabled	X			X			X		X			
.AlarmDscEnabled	X			X			X		X			
.AlarmDscInhibitor	X			X			X		X			
.AlarmDscUnAckCount	X			X			X		X			
.AlarmEnabled	X	X	X	X	X	X	X	X	X			
.AlarmGroup**											X	
.AlarmGroupSel **											X	
.AlarmHiDisabled***		X	X		X	X		X	X			
.AlarmHiEnabled***		X	X		X	X		X	X			
.AlarmHiHiDisabled***		X	X		X	X		X	X			
.AlarmHiHiEnabled***		X	X		X	X		X	X			
.AlarmHiHiInhibitor***		X	X		X	X		X	X			
.AlarmHiInhibitor***		X	X		X	X		X	X			
.AlarmLimit **											X	
.AlarmLoDisabled***		X	X		X	X		X	X			
.AlarmLoEnabled***		X	X		X	X		X	X			
.AlarmLoInhibitor***		X	X		X	X		X	X			
.AlarmLoLoDisabled***		X	X		X	X		X	X			
.AlarmLoLoEnabled***		X	X		X	X		X	X			
.AlarmLoLoInhibitor***		X	X		X	X		X	X			
.AlarmMajDevDisabled***		X	X		X	X		X	X			
.AlarmMajDevEnabled***		X	X		X	X		X	X			
.AlarmMajDevInhibitor***		X	X		X	X		X	X			
.AlarmMinDevDisabled***		X	X		X	X		X	X			
.AlarmMinDevEnabled***		X	X		X	X		X	X			
.AlarmMinDevInhibitor***		X	X		X	X		X	X			
.AlarmName **											X	
.AlarmOprName **											X	
.AlarmOprNode **											X	
.AlarmPri **											X	

标记名类型	内存型			I/O			间接		其它			
	离散型	整型	实型	离散型	整型	实型	离散型	模拟	报警组	Hist Trend	分布式报警对象/控件	Tag ID
.AlarmProv **											X	
.AlarmROC		X	X		X	X		X	X			
.AlarmROCCount		X	X		X	X		X	X			
.AlarmROCDisabled***		X	X		X	X		X	X			
.AlarmROCEnabled***		X	X		X	X		X	X			
.AlarmROCInhibitor***		X	X		X	X		X	X			
.AlarmROCUnAckCount		X	X		X	X		X	X			
.AlarmState **											X	
.AlarmTime **											X	
.AlarmTotalCount	X	X	X	X	X	X	X	X	X			
.AlarmType **											X	
.AlarmUnAckCount	X	X	X	X	X	X	X	X	X			
.AlarmUserDefNum1***	X	X	X	X	X	X	X	X	X			
.AlarmUserDefNum2***	X	X	X	X	X	X	X	X	X			
.AlarmUserDefStr***	X	X	X	X	X	X	X	X	X			
.AlarmValDeadband		X	X		X	X		X				
.AlarmValue **											X	
.AlarmValueCount		X	X		X	X		X	X			
.AlarmValueUnAckCount		X	X		X	X		X	X			
.Caption											X	
.ChartLength										X		
.ChartStart										X		
.Comment	X	X	X	X	X	X	X	X	X			
.DevTarget		X	X		X	X		X				
.DisplayMode										X		
.Enabled	X	X	X	X	X	X	X	X				
.EngUnits *		X	X		X	X		X				
.Freeze											X	
.HiHiLimit		X	X		X	X		X				
.HiHiSet **		X	X		X	X		X				
.HiHiStatus		X	X		X	X		X				
.HiLimit		X	X		X	X		X				
.HiSet **		X	X		X	X		X				
.HiStatus		X	X		X	X		X				
.ListChanged											X	
.ListCount											X	
.ListIndex											X	
.LoLimit		X	X		X	X		X				
.LoLoLimit **		X	X		X	X		X				
.LoLoSet		X	X		X	X		X				
.LoLoStatus		X	X		X	X		X				
.LoSet **		X	X		X	X		X				
.LoStatus		X	X		X	X		X				
.MajorDevPct		X	X		X	X		X				
.MajorDevSet **		X	X		X	X		X				
.MajorDevStatus		X	X		X	X		X				
.MaxEU		X	X		X	X		X				
.MaxRange										X		
.MaxRaw *					X	X		X				
.MinEU		X	X		X	X		X				
.MinorDevPct		X	X		X	X		X				
.MinorDevSet **		X	X		X	X		X				
.MinorDevStatus		X	X		X	X		X				
.MinRange										X		
.MinRaw *					X	X		X				
.Name	X	X	X	X	X	X	X	X	X	X	X	X
.NewIndex					X						X	
.NextPage											X	
.Normal	X	X	X	X	X	X	X	X	X			
.NumAlarms											X	
.OffMsg *	X			X			X					
.OnMsg *	X			X			X					
.PageNum											X	
.Pen1 through .Pen8										X		
.PendingUpdates											X	
.PrevPage											X	
.PriFrom											X	
.PriTo											X	
.ProviderReq											X	
.ProviderRet											X	
.Quality	X	X	X	X	X	X	X	X				
.QualityLimit *	X	X	X	X	X	X	X	X				
.QualityLimitstring *	X	X	X	X	X	X	X	X				

标记名类型	内存型			I/O			间接		其它			
	离散型	整型	实型	离散型	整型	实型	离散型	模拟	报警组	Hist Trend	分布式报警对象/控件	Tag ID
.QualityStatus *	X	X	X	X	X	X	X	X				
.QualityStatusString *	X	X	X	X	X	X	X	X				
.QualitySubstatus *	X	X	X	X	X	X	X	X				
.QualitySubstatusString *	X	X	X	X	X	X	X	X				
.QueryState											X	
.QueryType											X	
.RawValue *					X	X		X				
.ReadOnly											X	
.Reference				X	X	X	X	X				
.ReferenceComplete				X	X	X	X	X				
.ROCPct		X	X		X	X		X				
.ROCSet **		X	X		X	X		X				
.ROCStatus		X	X		X	X		X				
.ScooterLockLeft										X		
.ScooterLockRight										X		
.ScooterPosLeft										X		
.ScooterPosRight										X		
.Successful											X	
.SuppressRetain											X	
.TagID										X		
.TimeDate *	X	X	X	X	X	X	X	X				
.TimeDateString	X	X	X	X	X	X	X	X				
.TimeDateTime	X	X	X	X	X	X	X	X				
.TimeDay *	X	X	X	X	X	X	X	X				
.TimeHour *	X	X	X	X	X	X	X	X				
.TimeMinute *	X	X	X	X	X	X	X	X				
.TimeMonth *	X	X	X	X	X	X	X	X				
.TimeMsec *	X	X	X	X	X	X	X	X				
.TimeSecond *	X	X	X	X	X	X	X	X				
.TimeTime *	X	X	X	X	X	X	X	X				
.TimeTimeString	X	X	X	X	X	X	X	X				
.TimeYear *	X	X	X	X	X	X	X	X				
.TopIndex											X	
.TotalPages											X	
.UnAck	X	X	X	X	X	X	X	X	X			
.UpdateCount										X		
.UpdateinProgress										X		
.UpdateTrend										X		
.Value(Tagname)	X	X	X	X	X	X	X	X	X			X
.Value(Windows Control)											X	
.Visible											X	

.Ack 报警

监控所有类型的本地报警的报警确认状态。

用法

Tagname.**Ack=1**

参数	描述
Tagname	任意离散、整型或实型标记名、间接模拟标记名或报警组。

备注

此点域设置为 1 可确认与指定标记名/组关联的任何未确认报警。当指定的标记名为报警组时，指定组中与标记名关联的所有未确认报警均将得到确认。如果指定的标记名是任何报警组之外的类型，则只有与该标记名关联的未确认报警才会得到确认。将此点域设为 1 以外的值没有意义，其结果未定义。

数据类型

离散型（读/写）

有效值

1

实例

下面的语句确认与标记名“**Tag1**”关联的报警。

```
Tag1.Ack=1;
```

下面的例子用于确认报警组 **PumpStation** 内的所有未确认报警。

```
PumpStation.Ack=1;
```

注意：. Ack 有一个逆向点域叫 **.Unack**。当出现未确认报警时，**.UnAck** 将置为 0，**.UnAck** 随后可用于动画链接或条件脚本中，针对未确认报警触发电铃。

参见

.Alarm, .UnAck, .AckDev, AckROC, .AckDSC, .AckValue, .AlarmAckModel

.AckDev 报警

监控本地偏差报警的报警确认状态。

用法

`Tagname.AckDev=1`

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

此点域设置为 1 可确认与指定标记名/组关联的任何未确认偏差报警。当指定的标记名为报警组时，指定组中与标记名关联的所有未确认偏差报警均将得到确认。将此点域设为 1 以外的值没有意义，其结果未定义。

数据类型

离散型（读/写）

有效值

1

实例

下面的语句确认与标记名“**Tag1**”关联的偏差报警。

```
Tag1.AckDev=1;
```

下面的例子用于确认报警组 **PumpStation** 内的所有未确认偏差报警。

```
PumpStation.AckDev=1;
```

参见

.Alarm, .AlarmDev, .Ack, .UnAck, .AckDSC, .AckROC, .AckValue, .AlarmAckModel

.AckDsc 报警

监控本地离散报警的报警确认状态。

用法 *Tagname*`.AckDsc=1`

参数	描述
Tagname	任意间接离散标记名或报警组。

备注 此点域设置为 1 可确认与指定标记名/组关联的任何未确认离散报警。当指定的标记名为报警组时，指定组中与标记名关联的所有未确认离散报警均将得到确认。如果指定的标记名是任何报警组之外的类型，则只有与该标记名关联的未确认离散报警才会得到确认。将此点域设为 1 以外的值没有意义，其结果未定义。

数据类型 离散型（读/写）

有效值 1

实例 下面的语句确认与标记名“**Tag1**”关联的离散报警。

```
Tag1.AckDsc=1;
```

下面的例子用于确认报警组 **PumpStation** 内的所有未确认离散报警。

```
PumpStation.AckDsc=1;
```

参见 **.Alarm, .AlarmDSC, .Ack, .UnAck, .AckDev, .AckDSC, .AckROC, .AckValue, .AlarmAckModel**

.AckROC 报警

监控本地变化率报警的报警确认状态。

用法

`Tagname.AckROC=1`

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

此点域设置为 1 可确认与指定标记名/组关联的任何未确认变化率报警。当指定的标记名为报警组时，指定组中与标记名关联的所有未确认变化率报警均将得到确认。如果指定的标记名是任何报警组之外的类型，则只有与该标记名关联的未确认变化率报警才会得到确认。将此点域设为 1 以外的值没有意义，其结果未定义。

数据类型

离散型（读/写）

有效值

1

实例

下面的语句确认与标记名“**Tag1**”关联的变化率报警。

```
Tag1.AckROC=1;
```

下面的例子用于确认报警组 **PumpStation** 内的所有未确认变化率报警。

```
PumpStation.AckROC=1;
```

参见

.Alarm, .AlarmROC, .Ack, .UnAck, .AckDev, .AckDSC, .AckValue, .AlarmAckModel

.AckValue 报警

监控本地值报警的报警确认状态。

用法 *Tagname*.**AckValue=1**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 此点域设置为 1 可确认与指定标记名/组关联的任何未确认值报警。当指定的标记名为报警组时，指定组中与标记名关联的所有未确认值报警均将得到确认。如果指定的标记名是任意类型，则只有与该标记名关联的未确认值离散报警才会得到确认。将此点域设为 1 以外的值没有意义，其结果未定义。

数据类型 离散型（读/写）

有效值 1

实例 下面的语句确认与标记名“**Tag1**”关联的值报警。

```
Tag1.AckValue=1;
```

下面的例子用于确认报警组 **PumpStation** 内的所有未确认值报警。

```
PumpStation.AckValue=1;
```

间接报警组（使用组变量）可用于确认值报警。例如，使用下面的赋值语句：

```
StationAlarms.Name = "PumpStation";
```

此处，**StationAlarms** 定义为报警组类型标记名并与 **PumpStation** 关联。下面的语句类似于上面的例子，不同的是它用于确认与报警组标记名 **StationAlarms** 关联的报警组 **PumpStation** 内的未确认值报警。

```
StationAlarms.AckValue=1
```

参见 **.Alarm, .AlarmValue .Ack, .UnAck, .AckDev, .AckDSC, .AckROC, .AlarmAckModel**

.Alarm 报警

当指定标记名或报警组存在报警条件时，值等于 1。

用法 `Tagname.Alarm`

参数	描述
Tagname	任意离散、整型或实型标记名、间接模拟标记名或报警组。

备注 此只读点域通常等于 0。当指定标记名存在报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。如果指定的标记名是报警组名，则当该组内有任何标记名处于报警状态时，.Alarm 域将置为 1。.Alarm 有一个逆向点域叫 **.Normal**。

数据类型 离散型（只读）

有效值 0 或 1（离散型）

实例 下面的语句确认“Tag1”是否具有与之关联的活动报警：

```
IF (Tag1.Alarm == 1) THEN
```

如果报警组 **PumStation** 内具有活动报警，则会处理此 IF-THEN 语句的主体。

```
IF (PumpStation.Alarm == 1) THEN
   MyAlarmMessage="The pumping station currently has an
   ALARM!";
ENDIF;
```

此点域未链接到 **.Ack** 或 **.UnAck** 点域。因此，即使活动报警已获得确认，此点域仍等于 1。

参见 **.Ack, .Normal, Ack()**

.AlarmAccess 报警

返回与所选报警关联的标记名的访问名。必须通过单击摘要分布式报警显示来选定报警。

用法 **GetPropertyM(“Objectname.AlarmAccess”,*Tagname*);**

参数	描述
Objectname	分布式报警对象的名称。
Tagname	任意内存消息型标记名。

数据类型 内存消息型（只读）

实例 下面的语句可返回选定报警的标记名名称：

```
GetPropertyM("AlmObj_1.AlarmAccess",almAccess);
```

此处，*AlmObj_1* 是分布式报警对象名，*almAccess* 是一个内存消息型标记名，它包含与所选报警关联的标记名的访问名。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almAccess**，以便用于进一步的处理或显示。**almAccess** 将包含与所选报警关联的 I/O 标记名的访问名。

参见 **GetPropertyM(), .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue**

.AlarmAckModel 报警

如下监视与标记名关联的确认模型：

0 = 条件（缺省）
1 = 面向事件
2 = 扩展的摘要

用法

`Tagname.AlarmAckModel`

参数	描述
Tagname	任意离散、整型或实型标记名、间接模拟标记名或报警组。

备注

此点域缺省为 0（条件确认模型）。

数据类型

模拟型（只读）

有效值

0、1 或 2

实例

如果 **PumpStation** 为事件报警，则会处理以下 IF-THEN 语句的主体：

```
IF (PumpStation.AlarmAckModel == 1) THEN
   MyAlarmMessage="PumpStation is an Event alarm";
ENDIF;
```

参见

.Alarm, .Ack, .UnAck, .AckDev, .AckDSC, .AckROC

.AlarmClass 报警

返回与所选报警关联的标记名的类。必须通过单击摘要分布式报警显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmClass", Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

下面的语句返回与所选报警关联的报警类：

```
GetPropertyM("AlmObj_1.AlarmClass",almClass);
```

此处，*AlmObj_1* 是分布式报警对象的名称，*almClass* 是一个内存消息型标记名，它包含与所选报警关联的标记名的类。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almClass**，以便用于进一步的处理或显示。**almClass** 将包含与所选报警关联的报警类。

参见

GetPropertyM(), .AlarmAccess, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmComment 报警

读/写用于描述报警内容而非标记名内容的文本字符串。在新的应用程序中缺省条件下为空白。

但是，当旧的 InTouch 应用程序转换为 7.11 时，为保证反向兼容性，标记名注释将复制到 **AlarmComment** 点域中。

用法 1

```
Tagname.AlarmComment
```

参数	描述
Tagname	任意 InTouch 标记名或报警组。

数据类型

消息型（读/写）

用法 2

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmComment",Tagname
);
```

参数	描述
Tagname	内存消息型标记名
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。

数据类型

消息型（只读）

实例

用法 1：下面的语句会返回 **Tag1** 的报警注释，并将其置入内存消息型标记名 **mTag1** 中：

```
mTag1=Tag1.AlarmComment;
```

用法 2：下面的语句返回分布式报警对象 **AlmObj_1** 中选定的标记名的报警注释，并将其置入内存消息型标记名 **almComment** 中：

```
GetPropertyM(AlmObj_1.AlarmComment”, almComment);
```

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmDate 报警

返回与所选报警关联的标记名的日期。必须通过单击摘要分布式报警显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmDate", Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型 内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmDate",almDate);
```

此处，*AlmObj_1* 是分布式报警对象名，*almDate* 是一个内存消息型标记名，它包含与所选报警关联的标记名的日期。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almDate**，以便用于进一步的处理或显示。**almDate** 将包含与所选报警关联的报警日期。

参见 **GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue, .AlarmValue**

.AlarmDev 报警

当指定标记名或报警组存在偏差报警条件时，值等于 1。

用法

Tagname.**AlarmDev**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

此只读点域通常等于 0。当指定标记名存在偏差报警条件时，系统将其值设为 1。在偏差报警条件消失以前，此值将一直等于 1。如果指定的标记名是报警组名，则当该组内有任何标记名处于偏差报警状态时，.AlarmDev 将置为 1。

数据类型

离散型（只读）

有效值

0 或 1（离散型）

实例

下面的语句确认“**Tag1**”是否具有与之关联的活动偏差报警：

```
IF (Tag1.AlarmDev == 1) THEN
```

如果报警组 **PumStation** 内具有活动偏差报警，则会处理此 IF-THEN 语句的主体。

```
IF (PumpStation.AlarmDev == 1) THEN
   MyAlarmMessage="The pumping station currently has an
   ALARM!";
ENDIF;
```

此点域未链接到 **.Ack** 或 **.UnAck** 点域。因此，即使活动报警已获得确认，此点域仍等于 1。

参见

.Ack, .UnAck, .Alarm, .AckDev

.AlarmDevCount 报警

记录给定标记名或报警组上的活动偏差报警的总数。

用法 `Tagname.AlarmDevCount`

参数	描述
Tagname	任意整型或实型标记名，或报警组。

备注 包括副和主偏差报警计数。其中包括已确认和未确认的报警。对于非扩展摘要报警标记名，此计数始终为 1，但是计数可能会因报警组的不同而有所不同。

数据类型 模拟型（只读）

有效值 0 或任意正整数。

实例

```
ADC=Tag1.AlarmDevCount
```

此处，**Tag1** 是为偏差报警配置的模拟标记名。**ADC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的活动偏差（未确认和已确认）报警的总数。

参见 **.AlarmDSCCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmTotalCount, .AlarmDSCUnAckCount, .AlarmValueUnAckCount, .AlarmDevUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount**

.AlarmDevDeadband 报警

监控制副和主偏差报警的偏差死区百分比。

用法

Tagname.**AlarmDevDeadband**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画或脚本对此点域所做的更改。

数据类型

整型（读/写）

有效值

0 到 100（整数）

实例

下面的语句将偏差死区百分比改为 25%：

```
Tagname.AlarmDevDeadband=25;
```

参见

.AlarmValDeadband, .AlarmDev

.AlarmDevUnAckCount 报警

记录给定标记名或报警组上活动的未确认偏差报警总数。包括副和主偏差报警计数。

用法

Tagname.**AlarmDevUnAckCount**

参数	描述
Tagname	任意整型、实型标记名或报警组。

数据类型

模拟型（只读）

有效值

0 或任意正整数。

实例

```
ADUC = Tag1.AlarmDevUnAckCount
```

此处，**Tag1** 是为偏差报警配置的模拟标记名。**ADUC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的未确认偏差报警总数。

参见

.AlarmDevCount, .AlarmDSCCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmTotalCount, .AlarmDSCUnAckCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmDisabled 报警

禁用和（或）启用标记名或报警组的报警。

用法

Tagname.**AlarmDisabled**

参数	描述
Tagname	任意离散、整型或实型标记名、间接离散或间接模拟标记名，或报警组。

备注

当 **.AlarmDisabled** 设为 1 时，所有事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

当指定的标记名为报警组时，指定报警组中与标记名关联的所有报警均将被禁用。
此点域与 **.AlarmEnabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

0 = 启用报警（缺省）

1 = 禁用报警

实例

下面的实例启用标记名 **Tag1** 的报警。

```
Tag1.AlarmDisabled=0;
```

参见

.AlarmEnabled

.AlarmDsc 报警

当指定标记名或报警组存在离散报警条件时，值等于 1。

用法

Tagname.**AlarmDsc**

参数	描述
Tagname	任意离散、间接或离散型标记名，或报警组。

备注

此只读点域通常等于 0。当指定标记名存在离散报警条件时，系统将其值设为 1。在离散报警条件消失以前，此值将一直等于 1。如果指定的标记名是报警组名，则当该组内有任何标记名处于离散报警状态时，.AlarmDSC 域将置为 1。

数据类型

离散型（读/写）

有效值

0 或 1

实例

下面的语句确认“**Tag1**”是否具有与之关联的活动离散报警：

```
IF (Tag1.AlarmDsc == 1) THEN
MyAlarmMessage="The pumping station currently has an
   ALARM!";
ENDIF;
```

此点域未链接到 **.Ack** 或 **.UnAck** 点域。因此，即使活动报警已获得确认，此点域仍等于 1。

参见

.Ack, .UnAck, .Alarm, .AlarmDsc, .AckDsc

.AlarmDscCount 报警

记录给定标记名或报警组上的活动离散报警的总数。

用法

`Tagname.AlarmDscCount`

参数	描述
Tagname	任意离散型标记名或报警组。

备注

其中包括已确认和未确认的报警。对于非扩展摘要报警标记名，此计数始终为 1，但是计数可能会因报警组的不同而有所不同。

数据类型

模拟型（只读）

有效值

0 或任意正整数。

实例

```
ADC = Tag1.AlarmDSCCount
```

此处，**Tag1** 是为离散报警配置的模拟标记名。**ADC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的活动离散（未确认和已确认）报警的总数。

参见

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmTotalCount, .AlarmDscUnAckCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmDscDisabled 报警

指明标记名是否可产生离散报警。

用法 *Tagname*.**AlarmDscDisabled**

参数	描述
标记名	任意离散型标记名或报警组。

备注 当 **.AlarmDscDisabled** 设为 1 时，所有离散条件报警和事件均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmDscEnabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 0 = 启用报警（缺省）

1 = 禁用报警

实例 下面的实例启用 **Tag1** 中的离散报警：

```
Tag1.AlarmDscDisabled=0;
```

参见 **.AlarmDscEnabled**

.AlarmDscEnabled 报警

指明标记名是否可产生离散报警。

用法 *Tagname*.**AlarmDscEnabled**

参数	描述
Tagname	任意离散型标记名或报警组。

备注 当 **.AlarmDscEnabled** 设为 0 时，所有事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmDscDisabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 0 = 禁用报警

1 = 启用报警（缺省）

实例 下面的实例禁用 **Tag1** 中的离散报警：

```
Tag1.AlarmDscEnabled=0;
```

参见 **.AlarmDscDisabled**

.AlarmDscInhibitor 报警

返回指定给此标记名的离散报警的约束标记名（如果有）。

用法

Tagname.**AlarmDscInhibitor**

参数	描述
Tagname	任意离散型标记名或报警组。

备注

在 WindowMaker 配置，不能在运行时更改。

数据类型

消息型（只读）

实例

要使用 **.AlarmDSCInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmDscInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为离散报警返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmDscInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止

禁用 **AlarmedTag** 的离散报警

```
SomeIndirectTag = 0;
```

关闭报警禁止

可以产生 **AlarmedTag** 的离散报警

.AlarmDscUnAckCount 报警

记录给定标记名或报警组上活动的未确认离散报警总数。

用法 *Tagname*.**AlarmDscUnAckCount**

参数	描述
Tagname	任意离散型标记名或报警组。

数据类型 模拟型（只读）

有效值 所有整数和 0。

实例

```
ADUC = Tag1.AlarmDscUnACKCount
```

此处，**Tag1** 是为离散报警配置的模拟标记名。**ADUC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的未确认离散报警总数。

参见 **.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmValueCount, .AlarmROCCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount**

.AlarmEnabled 报警

启用和（或）禁用标记名或报警组的报警。

用法

`Tagname.AlarmEnabled`

参数	描述
Tagname	任意离散、整型或实型标记名、间接离散或间接模拟标记名，或报警组。

备注

当 **.AlarmEnabled** 设为 0 时，所有事件和报警被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

当指定的标记名为报警组时，将启用指定报警组中与标记名关联的所有报警。

此点域与 **.AlarmDisabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

0 = 禁用报警

1 = 启用报警（缺省）

实例

下面的实例禁用标记名 **Tag1** 的报警：

```
Tag1.AlarmEnabled=0;
```

参见

.AlarmDisabled

.AlarmGroup 报警

包含用于填入分布式报警显示对象的当前查询。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmGroup", Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时用于保存属性值的消息型标记名。

备注

此只读点域包含由指定分布式报警显示对象使用的当前报警查询。此查询可以是报警组列表或直接报警供应器引用。

数据类型

消息型（只读）

实例

下面的语句将分布式报警对象 "AlmObj_1" 使用的当前报警查询返回给消息型标记名 **CurrentQuery**：

```
GetPropertyM("AlmObj_1.AlarmGroup",CurrentQuery);
```

参见

GetPropertyM(), .AlarmGroupSel, .AlarmName

.AlarmGroupSel 报警

返回与所选报警关联的标记名的报警组。必须通过单击摘要分布式报警显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmGroupSel",
Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmGroup",almGroup);
```

此处，*AlmObj_1* 是分布式报警对象的名称，*almGroup* 是一个内存消息型标记名，它包含与所选报警关联的标记名的报警组。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almGroup**，以便用于进一步的处理或显示。**almGroup** 将包含与所选报警关联的报警组。

参见

GetPropertyM(), .AlarmGroup, .AlarmName

.AlarmHiDisabled 报警

禁用和（或）启用 High 条件的事件和报警。

用法 `Tagname.AlarmHiDisabled`

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmHiDisabled** 设为 1 时，所有 High 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmHiEnabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 1 = 禁用报警

0 = 启用报警（缺省）

实例 启用 Hi 报警：

```
Tag1.AlarmHiDisabled=0;
```

参见 **.AlarmHiEnabled, .AlarmDisabled**

.AlarmHiEnabled 报警

启用和（或）禁用 High 条件的事件和报警。

用法 *Tagname*.**AlarmHiEnabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmHiEnabled** 设为 0 时，所有 High 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmHiDisabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 0 = 禁用报警

1 = 启用报警（缺省）

实例 禁用 Hi 报警：

```
Tag1.AlarmHiEnabled=0;
```

参见 **.AlarmHiDisabled, .AlarmEnabled**

.AlarmHiHiDisabled 报警

禁用和（或）启用 HiHi 条件的事件和报警。

用法

`Tagname.AlarmHiHiDisabled`

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

当 **.AlarmHiHiDisabled** 设为 1 时，所有 HiHi 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmHiHiEnabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

1 = 禁用报警

0 = 启用报警（缺省）

实例

启用 HiHi 报警：

```
Tag1.AlarmHiHiDisabled=0;
```

参见

.AlarmHiHiEnabled, .AlarmDisabled

.AlarmHiHiEnabled 报警

启用和（或）禁用 HiHi 条件的事件和报警。

用法 *Tagname*.**AlarmHiHiEnabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmHiHiEnabled** 设为 0 时，所有 HiHi 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmHiHiDisabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 0 = 禁用报警

1 = 启用报警（缺省）

实例 禁用 HiHi 报警：

```
Tag1.AlarmHiHiEnabled=0;
```

参见 **.AlarmHiHiDisabled, .AlarmEnabled**

.AlarmHiHiInhibitor 报警

返回一个包含与 HiHi 报警条件关联的报警约束标记名的字符串。

用法

Tagname.**AlarmHiHiInhibitor**

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注

在 WindowMaker 中配置，不能在运行时更改。

数据类型

消息型（只读）

实例

要使用 **.AlarmHiHiInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmHiHiInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为 HiHi 报警限返回约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmHiHiInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的 HiHi 报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可以产生 **AlarmedTag** 的 HiHi 报警

参见

.AlarmHiInhibitor, .AlarmLoInhibitor, .AlarmLoLoInhibitor

.AlarmHiInhibitor 报警

返回与 High 报警条件关联的报警约束标记名。

用法

`Tagname.AlarmHiInhibitor`

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注

在 WindowMaker 中配置，不能在运行时更改。

数据类型

消息型（只读）

实例

要使用 **.AlarmHiInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmHiInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为 High 报警限返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmHiInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的 Hi 报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可以产生 **AlarmedTag** 的 Hi 报警

参见

.AlarmHiHiInhibitor, .AlarmLoInhibitor, .AlarmLoLoInhibitor

.AlarmLimit 报警

返回与所选报警关联的标记名的报警限。必须通过单击分布式报警摘要显示来选定报警。

用法

[ErrorNumber=]GetPropertyI("*ObjectName*.**AlarmLimit**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意消息型标记名。

数据类型

消息型（只读）

实例

```
GetPropertyI("AlmObj_1.AlarmLimit", almLimit);
```

此处，*AlmObj_1* 是分布式报警对象的名称，*almLimit* 是一个内存消息型标记名，它包含与所选报警关联的标记名的报警限。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almLimit**，以便用于进一步的处理或显示。**almLimit** 将包含与所选报警关联的报警限。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmLoDisabled 报警

禁用和（或）启用 Low 条件的事件和报警。

用法 *Tagname*.**AlarmLoDisabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmLoDisabled** 设为 1 时，所有 Low 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmLoEnabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 1 = 禁用报警

0 = 启用报警（缺省）

实例

```
Tag1.AlarmLoDisabled=0;
```

参见 **.AlarmDisabled, .AlarmEnabled, .AlarmLoEnabled**

.AlarmLoEnabled 报警

启用和（或）禁用 Low 条件的事件和报警。

用法

Tagname.**AlarmLoEnabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

当 **.AlarmLoEnabled** 设为 0 时，所有 Low 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmLoDisabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

0 = 禁用报警

1 = 启用报警（缺省）

实例

```
Tag1.AlarmLoEnabled=0;
```

参见

.AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled

.AlarmLoInhibitor 报警

返回与 Low 报警条件关联的报警约束标记名。

用法 *Tagname*.**AlarmLoInhibitor**

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注 在 WindowMaker 中配置，不能在运行时更改。

数据类型 消息型（只读）

实例 要使用 **.AlarmLoInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmLoInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为 Low 报警限返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmLoInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的 Lo 报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可以产生 **AlarmedTag** 的 Lo 报警

参见 **.AlarmHiInhibitor, .AlarmHiHiInhibitor, .AlarmLoLoInhibitor**

.AlarmLoLoDisabled 报警

禁用和（或）启用 LoLo 条件的事件和报警。

用法 *Tagname*.**AlarmLoLoDisabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmLoLoDisabled** 设为 1 时，所有 LoLo 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmLoLoEnabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 1 = 禁用报警

0 = 启用报警（缺省）

实例

```
Tag1.AlarmLoLoDisabled=0;
```

参见 **.AlarmDisabled, .AlarmEnabled, .AlarmLoLoEnabled**

.AlarmLoLoEnabled 报警

启用和（或）禁用 LoLo 条件的事件和报警。

用法 *Tagname*.**AlarmLoLoEnabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmLoLoEnabled** 设为 0 时，所有 LoLo 条件的事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmLoLoDisabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 0 = 禁用报警

1 = 启用报警（缺省）

实例

```
Tag1.AlarmLoLoEnabled=0;
```

参见 **.AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled**

.AlarmLoLoInhibitor 报警

返回与 LoLo 报警条件关联的报警约束标记名。

用法 *Tagname*.**AlarmLoLoInhibitor**

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注 在 WindowMaker 中配置，不能在运行时更改。

数据类型 消息型（只读）

实例 要使用 **.AlarmLoLoInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmLoLoInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为 LoLo 报警限返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmLoLoInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的 LoLo 报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可以产生 **AlarmedTag** 的 LoLo 报警

参见 **.AlarmHiInhibitor, .AlarmHiHiInhibitor, .AlarmLoInhibitor**

.AlarmMajDevDisabled 报警

禁用和（或）启用主偏差事件和报警。

用法 *Tagname*.**AlarmMajDevDisabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmMajDevDisabled** 设为 1 时，所有主偏差事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmMajDevEnabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 1 = 禁用报警

0 = 启用报警（缺省）

实例

```
Tag1.AlarmMajDevDisabled=0;
```

参见 **.AlarmDisabled, .AlarmEnabled, .AlarmMajDevEnabled**

.AlarmMajDevEnabled 报警

启用和（或）禁用主偏差事件和报警。

用法

`Tagname``.AlarmMajDevEnabled`

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

当 **.AlarmMajDevEnabled** 设为 0 时，所有主偏差事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmMajDevDisabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

0 = 禁用报警

1 = 启用报警（缺省）

实例

```
Tag1.AlarmMajDevEnabled=0;
```

参见

.AlarmDisabled, .AlarmEnabled, .AlarmMajDevDisabled

.AlarmMajDevInhibitor 报警

返回与主偏差报警条件关联的报警约束标记名。

用法 `Tagname.AlarmMajDevInhibitor`

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注 在 WindowMaker 中配置，不能在运行时更改。

数据类型 消息型（只读）

实例 要使用 **.AlarmMajDevInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmMajDevInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为主偏差报警限返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmMajDevInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的主偏差报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可以产生 **AlarmedTag** 的主偏差报警

参见 **.AlarmMinDevInhibitor**

.AlarmMinDevDisabled 报警

禁用和（或）启用副偏差事件和报警。

用法 *Tagname*.**AlarmMinDevDisabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmMinDevDisabled** 设为 1 时，所有副偏差事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmMinDevEnabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 1 = 禁用报警

0 = 启用报警（*缺省*）

实例

```
Tag1.AlarmMinDevDisabled=0;
```

参见 **.AlarmDisabled, .AlarmEnabled, .AlarmMinDevEnabled**

.AlarmMinDevEnabled 报警

启用和（或）禁用副偏差事件和报警。

用法 *Tagname*.**AlarmMinDevEnabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注 当 **.AlarmMinDevEnabled** 设为 0 时，所有副偏差事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmMinDevDisabled** 相同，但极性相反。

数据类型 离散型（读/写）

有效值 0 = 禁用报警

1 = 启用报警（缺省）

实例

```
Tag1.AlarmMinDevEnabled=0;
```

参见 **.AlarmDisabled, .AlarmEnabled, .AlarmMinDevDisabled**

.AlarmMinDevInhibitor 报警

返回与副偏差报警条件关联的报警约束标记名。

用法 *Tagname*.**AlarmMinDevInhibitor**

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注 在 WindowMaker 中配置，不能在运行时更改。

数据类型 消息型（只读）

实例 要使用 **.AlarmMinDevInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmMinDevInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为副偏差报警限返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmMinDevInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的副偏差报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可以产生 **AlarmedTag** 的副偏差报警

参见 **.AlarmMajDevInhibitor**

.AlarmName 报警

返回与所选报警关联的标记名的报警名。必须通过单击分布式报警摘要显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmName",
Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmName", almName);
```

此处，*AlmObj_1* 是分布式报警对象的名称，*almName* 是一个内存消息型标记名，它包含与所选报警关联的标记名的报警限名称。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almName**，以便用于进一步的处理或显示。**almName** 将包含与所选报警关联的报警名。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmOprName 报警

返回与所选报警关联的标记名的操作员姓名。必须通过单击分布式报警摘要显示来选定报警。

用法

[ErrorNumber=]GetPropertyM("*ObjectName*.**AlarmOprName**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmOprName", almOprName);
```

此处，*AlmObj_1* 是分布式报警对象的名称，*almOprName* 是一个内存消息型标记名，它包含与所选报警关联的标记名的操作员姓名。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almOprName**，以便用于进一步的处理或显示。**almOprName** 将包含与所选报警关联的报警操作员姓名。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmOprNode 报警

返回与所选报警关联的标记名的操作员节点。必须通过单击分布式报警摘要显示来选定报警。

用法

[ErrorNumber=]GetPropertyM("*ObjectName*.**AlarmOprNode**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

GetPropertyM("AlmObj_1.AlarmOprNode", almOprNode);

此处，*AlmObj_1* 是分布式报警对象的名称，*almOprNode* 是一个内存消息型标记名，它包含与所选报警关联的标记名的操作员节点。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almOprNode**，以便用于进一步的处理或显示。**almOprNode** 将包含与所选报警关联的报警操作员节点。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmPri 报警

返回与所选报警关联的标记名的优先级 (1-999)。必须通过单击分布式报警摘要显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyI("ObjectName.AlarmPri",
Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意模拟型标记名。

数据类型

内存消息型（只读）

有效值

1-999 的整数（只读）

实例

```
GetPropertyI("AlmObj_1.AlarmPri", almPri);
```

此处，*AlmObj_1* 是分布式报警对象名，*almPr* 是一个整型标记名，它包含与所选报警关联的标记名的优先级。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almPri**，以便用于进一步的处理或显示。**almPri** 将包含与所选报警关联的优先级。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmProv 报警

返回与所选报警关联的标记名的供应器。必须通过单击分布式报警摘要显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmProv", Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmProv", almProv);
```

此处，*AlmObj_1* 是分布式报警对象的名称，*almProv* 是一个内存消息型标记名，它包含与所选报警关联的标记名的供应器名称。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almProv**，以便用于进一步的处理或显示。**almProv** 将包含与所选报警关联的报警供应器。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmROC 报警

当指定标记名存在变化率报警条件时等于 1。

用法

Tagname.**AlarmROC**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

此只读点域通常等于 0。当指定标记名存在变化率报警条件时，系统将其值设为 1。在变化率报警条件消失以前，此值将一直等于 1。如果指定的标记名是报警组名，则当该组内有任何标记名处于变化率报警状态时，.Alarm 域将置为 1。

数据类型

离散型（只读）

有效值

0 或 1（离散型）

实例

下面的语句确认“**Tag1**”是否具有与之关联的活动变化率报警：

```
IF (Tag1.AlarmROC == 1) THEN
```

如果报警组 **PumStation** 内具有活动变化率报警，则会处理此 IF-THEN 语句的主体。

```
IF (PumpStation.AlarmROC == 1) THEN
   MyAlarmMessage="The pumping station currently has an
   ALARM!";
ENDIF;
```

此点域未链接到 **.Ack** 或 **.UnAck** 点域。因此，即使活动变化率报警已获得确认，此点域仍等于 1。

参见

.Ack, .AckROC, .Alarm, .AlarmROCEnabled, .AlarmROCDisabled

.AlarmROCCount 报警

记录给定标记名或报警组上活动变化率报警的总数。其中包括已确认和未确认的报警。对于非扩展摘要报警标记名，此计数始终为 1，但是计数可能会因报警组的不同而有所不同。此点域为只读点域。

用法 *Tagname*.**AlarmROCCount**

参数	描述
Tagname	任意整型、实型标记名或报警组。

备注

数据类型 整型（只读）

有效值 0 或任意正整数。

实例 **ARC = Tag1.AlarmROCCount**

此处，**Tag1** 是为变化率报警配置的模拟标记名。**ARC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的活动变化率（未确认和已确认）报警的总数。

参见 **.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount**

.AlarmROCDisabled 报警

禁用和（或）启用变化率事件和报警。

用法

Tagname.**AlarmROCDisabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

当 **.AlarmROCDisabled** 设为 1 时，所有变化率事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmROCEnabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

1 = 禁用报警

0 = 启用报警（*缺省*）

实例

```
Tag1.AlarmROCDisabled=0;
```

参见

.AlarmDisabled, .AlarmEnabled, .AlarmROCEnabled

.AlarmROCEnabled 报警

启用和（或）禁用变化率事件和报警。

用法

Tagname.**AlarmROCEnabled**

参数	描述
Tagname	任意整型或实型标记名、间接模拟标记名或报警组。

备注

当 **.AlarmROCEnabled** 设为 0 时，所有变化率条件事件和报警均将被忽略。它们不会保存到缓冲区，也不会写入磁盘中。因此应尽可能地重新激活事件/报警以避免数据丢失。

此点域与 **.AlarmROCDisabled** 相同，但极性相反。

数据类型

离散型（读/写）

有效值

0 = 禁用报警

1 = 启用报警（*缺省*）

实例

```
Tag1.AlarmROCEnabled=0;
```

参见

.AlarmDisabled, .AlarmEnabled, .AlarmROCDisabled

.AlarmROCInhibitor 报警

返回与变化率报警条件关联的报警约束标记名。

用法 *Tagname*.**AlarmROCInhibitor**

参数	描述
Tagname	包含报警或报警组的模拟标记名。

备注 在 WindowMaker 中配置，不能在运行时更改。

数据类型 消息型（只读）

实例 要使用 **.AlarmROCInhibitor** 点域，可将 **.Name** 设为等于 **.AlarmROCInhibitor** 标记名值的间接标记名，然后使用间接标记名的值。下面的语句将为变化率报警返回报警约束标记名的名称（假定 **SomeIndirectTag** 为模拟间接标记名）：

```
SomeIndirectTag.Name = AlarmedTag.AlarmROCInhibitor;
```

报警标记名的禁止状态可以通过如下设置间接标记名的值来进行控制：

```
SomeIndirectTag = 1;
```

打开报警禁止
禁用 **AlarmedTag** 的变化率报警

```
SomeIndirectTag = 0;
```

关闭报警禁止
可产生 **AlarmedTag** 的变化率报警

.AlarmROCUnAckCount 报警

记录给定标记名或报警组上当前未确认的报警总数。

用法

Tagname.**AlarmROCUnAckCount**

参数	描述
Tagname	任意整型、实型标记名或报警组。

数据类型

整型（只读）

有效值

0 或任意正整数。

实例

```
ARUC = Tag1.AlarmROCUnACKCount
```

此处，**Tag1** 是为变化率报警配置的模拟标记名。**ARUC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的未确认变化率报警总数。

参见

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCCount, .AlarmUnAckCount

.AlarmState 报警

返回与所选报警关联的标记名的状态。必须通过单击分布式报警摘要显示来选定报警。

用法

`[ErrorNumber=]GetPropertyM("`*`ObjectName`*`.`**`AlarmState`**`", `*`Tagname`*`);`

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmState", almState);
```

此处，*AlmObj_1* 是分布式报警对象名，*almState* 是一个内存消息型标记名，它包含与所选报警关联的标记名的报警状态。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almState**，随后便可用于进一步的处理或显示。**almState** 将包含与所选报警关联的报警状态。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmTime, .AlarmType, .AlarmValue

.AlarmTime 报警

返回与所选报警关联的标记名的时间。必须通过单击分布式报警摘要显示来选定报警。

用法

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmTime",
Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmTime", almTime);
```

此处，*AlmObj_1* 是分布式报警对象名，*almTime* 是一个内存消息型标记名，它包含与所选报警关联的标记名的时间。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almTime**，以便用于进一步的处理或显示。**almTime** 将包含与所选报警关联的报警时间。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmType, .AlarmValue

.AlarmTotalCount 报警

记录给定标记名或报警组上活动报警的总数。

用法 *Tagname*.**AlarmTotalCount**

参数	描述
Tagname	所有类型的标记名。

备注 包括值、偏差、变化率和离散报警的计数。其中包括已确认和未确认的报警。

数据类型 整型（只读）

有效值 0 或任意正整数。

实例

```
ATC = Tag1.AlarmTotalCount
```

此处，**Tag1** 是为报警配置的模拟标记名。**ATC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的所有活动报警（未确认和已确认）的总数。

参见 **.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDSCCount, .AlarmDSCUnAckCount, .AlarmValueCount, .AlarmUnAckCount, .AlarmValueUnAckCount, .AlarmROCCount, .AlarmROCUnACkCount**

.AlarmType 报警

返回与所选报警关联的标记名的类型。必须通过单击分布式报警摘要显示来选定报警。

用法

[ErrorNumber=]GetPropertyM("*ObjectName*.**AlarmType**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	任意内存消息型标记名。

数据类型

内存消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmType", almType);
```

此处，*AlmObj_1* 是分布式报警对象名，*almType* 是一个内存消息型标记名，它包含与所选报警关联的标记名的类型。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almType**，以便用于进一步的处理或显示。**almType** 将包含与所选报警关联的报警类型。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmValue

.AlarmUnAckCount 报警

记录给定标记名或报警组上活动的未确认报警总数。

用法

```
Tagname.AlarmUnAckCount
```

参数	描述
Tagname	所有类型的标记名。

备注

包括值、偏差、变化率和离散报警的计数。

数据类型

整型（只读）

有效值

0 或任意正整数。

实例

```
AUC = Tag1.AlarmUnACKCount
```

此处，**Tag1** 是为报警配置的模拟或离散型标记名。AUC 也是一个模拟标记名，它可以取得 **Tag1** 中存在的未确认报警总数。

参见

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCCount, .AlarmROCUnACkCount

.AlarmUserDefNum1 报警

如果您设置此点域，则它会添加到报警数据库中由 Alarm Logger 记录的每个报警信息中（**.AlarmUserDefNum1** 对应于数据库字段 User1）。您可以在 SELECT 语句中使用自定义列来选择用于数据库操作的特定报警集合。例如，如果 **$System.AlarmUserDefNum1** 设为“批号”，并且它会随批次的更改而更改，则选择数据库字段 User1 时可选择特定批次的报警。

用法 `Tagname.AlarmUserDefNum1`

参数	描述
Tagname	任意离散、整型或实型标记名、间接离散或间接模拟标记名，或报警组。

备注 此自定义点域可用于各种标记名，特别是离散标记名、模拟标记名和报警组（无论是否已定义报警）。您可以针对任意的单个标记名、报警组或父报警组，设置这些项目的全部、一部分或不设置。

数据类型 模拟型（读/写）

有效值 0.0 或不设置（缺省）

实例 此点域的值会附加在报警上，但只限于已设置值的情况下（例如通过脚本或插入来设置）。

```
$System.AlarmUserDefNum1 = 4;
GroupA.AlarmUserDefNum1 = 27649;
```

上述所有实例均使用常数值。但是，您可以编写 InTouch QuickScrip，将另一标记名的值复制到自定义点域中。您也可以使用 *PtAcc* 来设置或检查，或使用 InTouch 作为 I/O Server 来取得或设置点域值。

从理论上讲，当报警通知发送到分布式报警系统时，系统将使用最低级的设置。即是说，如果标记名的 **.AlarmUserDefNum1** 已设置某些值，则系统会使用该设置进行报警记录。但是，如果标记名没有进行任何设置，WindowViewer 会检查标记名的报警是否进行设置，如此逐层升级直至达到根组 **$System**。如果在任意级别未发现设置，则报警记录项将保留空白（如为数字，保留零；如为字符串，则保留空白字符串）。

注意：此分层结构搜索根据不同的项目进行处理。因此，如果标记名已设置 **.AlarmUserDefNum2** 但未设置 **.AlarmUserDefNum1**，但其父组已设置 **.AlarmUserDefNum1**，则报标记名将从其父组继承 **.AlarmUserDefNum1** 的设置。

参见 **.AlarmUserDefNum2, .AlarmUserDefStr**

.AlarmUserDefNum2 报警

如果您设置此点域，则它会添加到报警数据库中由 AlarmLogger 记录的各个报警信息中（**.AlarmUserDefNum2** 对应于数据库字段 User2）。您可以在 SELECT 语句中使用自定义列来选择用于数据库操作的特定报警集合。例如，如果 **$System.AlarmUserDefNum2** 设为“批号”，并且它会随批次的更改而更改，则选择数据库字段 User2 时可选择特定批次的报警。

用法

Tagname.**AlarmUserDefNum2**

参数	描述
Tagname	任意离散、整型或实型标记名、间接离散或间接模拟标记名，或报警组。

备注

此自定义点域可用于各种标记名，特别是离散标记名、模拟标记名和报警组（无论是否已定义报警）。您可以针对任意的单个标记名、报警组或父报警组，设置这些项目的全部、一部分或不设置。

数据类型

模拟型（读/写）

有效值

0.0 或不设置（缺省）

实例

此点域的值会附加在报警上，但只限于已设置值的情况下（例如通过脚本或插入来设置）。

```
$System.AlarmUserDefNum2 = 4;
GroupA.AlarmUserDefNum2 = 27649;
```

上述所有实例均使用常数值。但是，您可以编写 InTouch QuickScrip，将另一标记名的值复制到自定义点域中。您也可以使用 *PtAcc* 来设置或检查，或使用 InTouch 作为 I/O 服务程序来取得或设置点域值。

从理论上讲，当报警通知发送到分布式报警系统时，系统将使用最低级的设置。即是说，如果标记名的 **.AlarmUserDefNum2** 已设置某些值，则系统会使用该设置进行报警记录。但是，如果标记名没有进行任何设置，WindowViewer 会检查标记名的报警是否进行设置，如此逐层升级直至达到根组 **$System**。如果在任意级别未发现设置，则报警记录项将保留空白（如为数字，保留零；如为字符串，则保留空白字符串）。

注意：此分层结构搜索根据不同的项目进行处理。因此，如果标记名已设置 **.AlarmUserDefNum1** 但未设置 **.AlarmUserDefNum2**，但其父组已设置 **.AlarmUserDefNum2**，则报标记名将从其父组继承 **.AlarmUserDefNum2** 的设置。

参见

.AlarmUserDefNum1, .AlarmUserDefStr

.AlarmUserDefStr 报警

如果您设置此点域，则它会添加到报警数据库中由 Alarm Logger 记录的每个报警信息中（**.AlarmUserDefStr** 对应数据库字段 User3）。您可以在 SELECT 语句中使用自定义列来选择用于数据库操作的特定报警集合。例如，如果 **$System.AlarmUserDefStr** 设为“批字符串”，并且它会随批次的更改而更改，则选择数据库字段 User3 时可选择特定批次的报警。

用法

Tagname.**AlarmUserDefStr**

参数	描述
Tagname	任意离散、整型或实型标记名、间接离散或间接模拟标记名，或报警组。

备注

此自定义点域可用于各种标记名，特别是离散标记名、模拟标记名和报警组（无论是否已定义报警）。您可以针对任意的单个标记名、报警组或父报警组，设置这些项目的全部、一部分或不设置。

数据类型

消息型（读/写）

有效值

NULL 及任何无效字符串。

实例

此点域的值会附加在报警上，但只限于已设置值的情况下（例如通过脚本或插入来设置）。

```
Tag04.AlarmUserDefStr = "Joe";
```

此实例使用常数值。但是，您可以编写 InTouch QuickScrip，将另一标记名的值复制到自定义点域中。您也可以使用 *PtAcc* 来设置或检查，或使用 InTouch 作为 I/O 服务程序来取得或设置点域值。

从理论上讲，当报警通知发送到分布式报警系统时，系统将使用最低级的设置。即是说，如果标记名的 **.AlarmUserDefStr** 已设置某些值，则系统会使用该设置进行报警记录。但是，如果标记名没有进行任何设置，WindowViewer 会检查标记名的报警是否进行设置，如此逐层升级直至达到根组 **$System**。如果在任意级别未发现设置，则报警记录项将保留空白（如为数字，保留零；如为字符串，则保留空白字符串）。

请注意此分层结构搜索根据不同的项目进行处理。因此，如果标记名已设置 **.AlarmUserDefNum1** 但未设置 **.AlarmUserDefNum2**，但其父组已设置 **.AlarmUserDefStr**，则报警记录将使用该设置。

参见

.AlarmUserDefNum1, .AlarmUserDefNum2

.AlarmValDeadband 报警

监控报警死区值。

用法 *Tagname*.**AlarmValDeadband**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画或脚本对此点域所做的更改。

数据类型 模拟型（读/写）

有效值 必须位于指定标记名的配置值范围内。

实例 下面的语句将标记名“**Tag1**”的死区值改为 25：

```
Tag1.AlarmValDeadband=25;
```

参见 **.AlarmDevDeadband**

.AlarmValue 报警

返回与所选报警关联的标记名的报警值。必须通过单击摘要分布式报警显示来选定报警。

用法

[ErrorNumber=]GetPropertyM("*ObjectName*.**AlarmValue**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	任意消息型标记名。

数据类型

消息型（只读）

实例

```
GetPropertyM("AlmObj_1.AlarmValue", almValue);
```

此处，*AlmObj_1* 是分布式报警对象名，*almValue* 是一个内存消息型标记名，它包含与所选报警关联的标记名的报警值。

如果在触动按钮 QuickScript 中使用此点域，则当操作员按下按钮时，此语句会将值返回 **almValue**，以便用于进一步的处理或显示。**almValue** 将包含与所选报警关联的报警值。

参见

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType

.AlarmValueCount 报警

记录给定标记名或报警组上活动值报警的总数。

用法 *Tagname***.AlarmValueCount**

参数	描述
Tagname	任意整型或实型标记名，或报警组。

备注 包括 HiHi、Hi、Lo 和 LoLo 报警计数。其中包括已确认和未确认的报警。对于非扩展摘要报警标记名，此计数不会超过 1，但是计数可能会因报警组的不同而有所不同。

数据类型 整型（只读）

有效值 0 或任意正整数。

实例

```
AVC = Tag1.AlarmValueCount
```

此处，**Tag1** 是为整型报警配置的模拟标记名。**AVC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的所有报警值的总数。

参见 **.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmROCCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount**

.AlarmValueUnAckCount 报警

记录给定标记名或报警组上活动的未确认值报警总数。包括 HiHi、Hi、Lo 和 LoLo 报警计数。

用法 *`Tagname`*`.`**`AlarmValueUnAckCount`**

参数	描述
Tagname	任意整型或实型标记名，或报警组。

数据类型 整型（只读）

有效值 0 或任意正整数。

实例 **`AVUC = Tag1.AlarmValueUnAckCount`**

此处，**Tag1** 是为整型报警配置的模拟标记名。**AVUC** 也是一个模拟标记名，它可以取得 **Tag1** 中存在的所有未确认报警值的总数。

参见 **.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmROCCount, .AlarmTotalCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmUnAckCount**

.Caption 窗口控件

确定由复选框显示的“消息”。

用法

```
[ErrorNumber=]GetPropertyM ("ControlName.Caption", Tagname);
```

```
[ErrorNumber=]GetPropertyM ("ControlName.Caption", "Message");
```

参数	描述
ControlName	Name of the windows control.例如 *ChkBox_4*。
Tagname	保存所请求属性的消息型标记名。
"Message"	用引号括起的消息型字符串。

备注 此属性在开发和运行时读/写。

数据类型 消息型（读/写）

应用于 复选框。

实例 此语句将复选框对象“CheckBox_1”的标题设为“Blue Paint Option”。

```
SetPropertyM("CheckBox_1.Caption","Blue Paint Option");
```

参见 **GetPropertyM(), SetPropertyM()**

.ChartLength 历史

控制历史趋势图表上显示的时间长度（以秒为单位）。

用法 *Tagname*.**ChartLength**

参数	描述
Tagname	任意历史趋势标记名。

备注 此读/写点域用于设置（或确认）历史趋势图表的图表长度值。图表长度用秒表示。长度定义为当前历史趋势图表显示的时间量。具体地说，从历史趋势图表中获取图表长度的计算为：

ChartLength=（图表右端的日期/时间标签）-（图表左端的日期/时间标签）;

由于日期/时间标签用 70 年 1 月 1 日午夜以来的秒数表示，所以计算结果为“图表左右端之间显示的时间的秒数差”。

每当从 **.ChartLength** 加或减时，切记您使用的是秒数。所以，如果您要从当前的 **.ChartLength** 减去“2 小时”，您必须在计算前将“2 小时”转换为秒。例如：

(2 小时) * (60 分钟/小时) * (60 秒/分钟) = 7200 秒。

数据类型 整型（读/写）

有效值 任何正整数

实例 下面的语句强制图表跨度为一小时：

```
HtTagname.ChartLength=3600 {60 minutes * 60 seconds/minute};
```

下面的语句则使图表向左滚动一半：

```
HtTagname.ChartStart=HtTagname.ChartStart -
HtTagname.ChartLength / 2;
```

下面的语句使图表向左滚动 10%：

```
HtTagname.ChartStart=HtTagname.ChartStart - (.10 *
HtTagname.ChartLength);
```

参见 **.ChartStart**

.ChartStart 历史

控制历史趋势图表的开始日期/时间标签。

用法 *Tagname*.**ChartStart**

参数	描述
Tagname	任意历史趋势标记名。

备注 此读/写点域用于设置（或确认）历史趋势图表的开始（左端）日期/时间标签值。**.ChartStart** 域以自 70 年 1 月 1 日午夜以来的秒数表示，“开始”定义为历史趋势图表左端的第一个日期/时间标签。

数据类型 整型（读/写）

有效值 任何正整数

实例 下面的语句使图表向右滚动 1 分钟：

```
HtTagname.ChartStart=HtTagname.ChartStart + 60;
```

参见 **.ChartLength**

.Comment 标记名

包含标记名字典中的标记名“注释”字段值。

用法

`Tagname.Comment`

参数	描述
Tagname	任意标记名。

备注

如果此域在 WindowMaker 中为只读，可以在 WindowViewer 中更改其值（仅限于内存型）。一旦 WindowViewer 关闭并重新启动，将使用**注释**域的旧值。该域用于分布报警系统的报警注释。

数据类型

消息型（只读）

有效值

包括 1 到 50 个字符的任意字符串。有关字符串限制的信息，参见下面的附注。

注意：标记名的“**注释**”域只能在标记名字典中更改。在 WindowViewer 中输入 **.Comment** 域的值不会写入标记名数据库。

实例

下面的语句通过加入括标记名的名称及其“**注释**”域的值来组成一个字符串（消息型标记名的内容）。

```
OperatorMessage=MyTag.Name + " has a comment of: " +
MyTag.Comment;
```

注意：此域会写入运行时数据库，但不会保存到标记名字典中。

.DevTarget 报警

监控副和主偏差报警的目标。

用法

Tagname.**DevTarget**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画链接或 QuickScript 对此点域所做的更改。

数据类型

模拟型（读/写）

有效值

必须位于指定标记名的配置值范围内。

实例

下面的语句将标记名 **MyTag** 的偏差目标值设为 500。

```
MyTag.DevTarget=500;
```

参见

.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevSet, .MajorDevStatus

.DisplayMode 历史

确定在趋势上显示值的方法。

用法

Tagname.**DisplayMode**

参数	描述
Tagname	任意历史趋势标记名。

数据类型 模拟型（读/写）

有效值 1＝显示每个采样周期的最小/最大值（缺省）。

2 = 在“散点”图中显示每个采样周期的平均值。

3 = 在“条形”图中显示每个采样周期的平均值。

实例 下面的语句指示与 **MyHistTrendTag** 关联的历史趋势显示一个条形图。

```
MyHistTrendTag.DisplayMode=3;
```

参见 **.ChartLength, .ChartStart**

.Enabled 窗口控件

确定控件对象能否响应用户生成的事件。

用法

```
[ErrorNumber=] GetPropertyD("ControlName.Enabled", Tagname);
```

```
[ErrorNumber=] SetPropertyD("ControlName.Enabled", Discrete);
```

参数	描述
ControlName	窗口控件名，例如： *ChkBox_4*
Tagname	保存所请求属性的离散型标记名。
Discrete	离散值： 0 = 禁用控件 1 = 启用控件 -或者- 当函数运行时，保存需写入值的离散型标记名。

备注 此属性在开发和运行时读/写。

数据类型 离散型（读/写）

应用于 文本框、列表框、组合框、复选框和单选钮。

实例 下面的语句禁用列表框对象： "ListBox_1"。

```
SetPropertyD("ListBox_1.Enabled",0);
```

参见 **GetPropertyD(), SetPropertyD()**

.EngUnits 标记名

“**工程单位**”域允许用户访问在标记名字典中指定的模拟标记名的工程单位。

用法

Tagname.**EngUnits**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

此域为文本域，它不影响比例、转换或标记名值的格式。

数据类型

消息型（读/写）

注意：写入此域的值不会保留。

有效值

包括 0 到 31 个字符的任意字符串。

实例

```
IF Temperature.EngUnits == "Celsius" THEN.
    CALL TempConvert(Temperature);{一个可用于在摄氏和华氏之间转换的
    QuickFunction}
ENDIF;
```

参见

.MinEU, .MaxEU, .MaxRange, .MinRange, .MinRaw, .MaxRaw, .RawValue

.Freeze 报警

读/写分布式报警显示对象的冻结状态。

用法

```
[ErrorNumber=]GetPropertyD("ObjectName.Freeze",Tagname");
[ErrorNumber=]SetPropertyD("ObjectName.Freeze",Tagname");
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	当函数运行时用于保存属性值的离散型标记名。

备注 读写型点域，包含或更改分布式报警显示对象的冻结状态。

数据类型 离散型（读/写）

有效值 0 = 关闭冻结

1 = 开启冻结

实例 下面的语句通过离散型标记名 **AlmFreeze** 设置 "**AlmObj_1**" 的"冻结"属性。

```
SetPropertyD("AlmObj_1.Freeze",AlmFreeze);
```

参见 **GetPropertyD(), SetPropertyD()**

.HiHiLimit 报警

监控值报警检查的 HiHi 报警限。

用法

Tagname.**HiHiLimit**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画链接或 QuickScript 对此点域所做的更改。

数据类型

模拟型（读/写）

有效值

必须位于指定标记名的配置值范围内。

实例

下面的语句将标记名 **MyTag** 的 HiHi 报警限增加 5：

```
MyTag.HiHiLimit=MyTag.HiHiLimit + 5;
```

参见

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.HiHiSet 报警

取决于模拟型标记名是否设置了 HiHi 报警限条件，返回值 0（假）或 1（真）。

用法

Tagname.**HiHiSet**

参数	描述
Tagname	任意模拟型标记名。

备注

此点域只能用于整型或实型标记名。

数据类型

离散型

有效值

1（真）或 0（假）

实例

下面的语句可用于“模拟输出”链接中，用于确定是否设置了标记名 **MyTag** 的 HiHi 报警限：

```
MyTag.HiHiSet 5;
```

{**如果未设置，此处返回 0；如果已设置，则返回 1。**}

参见

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiLimit, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.HiHiStatus 报警

确认是否存在 HiHi 报警限。

用法 *Tagname*.**HiHiStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 此只读点域通常等于 0。当指定标记名存在 HiHi 值报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

数据类型 离散型（只读）

有效值 0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例 当标记名 **MyTag** 的 **.HiHiStatus**（HiHi 报警）等于 1 时，将执行下面的 IF-THEN 语句。

```
IF (MyTag.HiHiStatus == 1) THEN
   OperatorMessage="MyTag has gone into HiHi Alarm";
ENDIF;
```

参见 **.Alarm, .AlarmValue, .Ack, .HiHiLimit, .HiHiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor**

.HiLimit 报警

监控值报警检查的 High 报警限。

用法 `Tagname.HiLimit`

参数	描述
Tagname	任意模拟型标记名。

备注 在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画链接或 QuickScript 对此点域所做的更改。

数据类型 模拟型（读/写）

有效值 必须位于指定标记名的配置值范围内。

实例

```
Temperature.HiLimit = 212;
```

参见 **.Alarm, .AlarmValue, .Ack, .HiStatus, .HiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiDisabled, .AlarmHiEnabled, .AlarmHiInhibitor**

.HiSet 报警

取决于模拟型标记名是否设置了 High 报警限，返回值 0（假）或 1（真）。

用法

Tagname.**HiSet**

参数	描述
Tagname	任意模拟型标记名。

备注

此点域只能用于整型或实型标记名。

数据类型

离散型

有效值

1（真）或 0（假）

实例

下面的语句可用于“模拟输出”链接中，用于确定是否设置了标记名 **MyTag** 的 High 报警限：

```
MyTag.HiSet;
```

{**如果未设置，此处返回 0；如果已设置，则返回 1。**}

参见

.Alarm, .AlarmValue, .Ack, .HiStatus, .HiLimit, .AlarmDisabled, .AlarmEnabled, .AlarmHiDisabled, .AlarmHiEnabled, .AlarmHiInhibitor

.HiStatus 报警

确认是否存在 High 报警。

用法 *Tagname*.**HiStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 此只读点域通常等于 0。当指定标记名存在 High 报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

数据类型 离散型（只读）

有效值 0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例

```
IF (MotorAmps.HiStatus == 1) THEN
   CALL PumpShutdown( );
ENDIF;
```

{可用于清除泵马达输出的 QuickFunction}

参见 **.Alarm, .AlarmValue, .Ack, .HiLimit, .HiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiDisabled, .AlarmHiEnabled, .AlarmHiInhibitor**

.ListChanged 报警

指明是否存在任何新报警或分布式报警对象的更新。

用法

```
[ErrorNumber=]GetPropertyD("ObjectName.ListChanged",Tagname)
;
```

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	当函数运行时用于保存属性值的离散型标记名。

备注

此只读点域包含有关是否需要在分布式报警显示对象中更新任何更改的信息。此属性会在读取时自动重置。

数据类型

离散型（只读）

有效值

0 = 显示对象没有新报警或更新。

1 = 显示对象存在更新。

实例

下面的语句返回：离散型标记名 **AlmDispStat** 的分布式报警对象 **AlmObj_1** 是否存在任何新报警或更新。

```
GetPropertyD("AlmObj_1.ListChanged",AlmDispStat);
```

参见

Get PropertyD()

.ListCount 窗口控件

确定列表框或组合框中的项目数。

用法

```
[ErrorNumber=]GetPropertyI("ControlName.ListCount",
Tagname);
```

参数	描述
ControlName	窗口控件名，例如，*ListBox_4*。
Tagname	包含列表内的项目整数计数的有效标记名。

备注 此属性只在运行时可用。

数据类型 整型（只读）

应用于 列表框和组合框。

实例 下面的语句检索列表框“ListBox_1”中的项目数，然后将该值写入内存整型标记名 **MyListBoxCount** 中。

```
GetPropertyI("ListBox_1.ListCount",MyListBoxCount);
```

参见 **GetPropertyI(), .ListIndex**

.ListIndex 窗口控件

确定列表中当前选定项的相应索引(*标记名*或*数字*)。索引是定义列表中的特定项目的数字。当使用列表框时，索引 -1 指示当前未选定任何项目。当使用组合框时，索引 -1 指示用户在控件的文本输入域中输入了新文本。

用法

```
[ErrorNumber=]GetPropertyI("ControlName.ListIndex", Tagname);
```

```
[ErrorNumber=]SetPropertyI("ControlName.ListIndex", Number);
```

参数	描述
ControlName	窗口控件名，例如，*ListBox_4*。
Tagname	包含列表内的项目整数计数的有效标记名。
Number	定义列表中的特定项目的索引号。

备注 此属性只在运行时可用。

数据类型 整型（读/写）

应用于 列表框和组合框。

实例 下面的语句检索列表框“Listbox_1”中当前选定项目的索引，然后将该值写入内存整型标记名 **MyListBoxIndex** 中。

```
GetPropertyI( "ListBox_1.ListIndex",MyListBoxIndex );
```

参见 **GetPropertyI(), SetPropertyI(), .NewIndex, .TopIndex**

.LoLimit 报警

监控值报警检查的下限。

用法

`Tagname.LoLimit`

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画链接或 QuickScript 对此点域所做的更改。

数据类型

模拟型（读/写）

有效值

必须位于指定标记名的配置值范围内。

实例

此语句将标记名 **MyTag** 的 Low 报警限降低 10：

```
MyTag.LoLimit=MyTag.LoLimit - 10;
```

参见

.Alarm, .AlarmValue, .Ack, .LoStatus, .LoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled, .AlarmLoEnabled, .AlarmLoInhibitor

.LoLoLimit 报警

监控值报警检查的 LoLo 报警限。

用法

Tagname.**LoLoLimit**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画链接或 QuickScript 对此点域所做的更改。

数据类型

模拟型（读/写）

有效值

必须位于指定标记名的配置值范围内。

实例

此语句将标记名 **MyTag** 的 LoLo 报警限降低 10：

```
MyTag.LoLoLimit=MyTag.LoLoLimit - 10;
```

参见

.Alarm, .AlarmValue, .Ack, .LoLoStatus, .LoLoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled, .AlarmLoLoEnabled, .AlarmLoLoInhibitor

.LoSet 报警

取决于模拟型标记名是否设置了 Low 报警限，返回值 0（假）或 1（真）。

用法

`Tagname.LoSet`

参数	描述
Tagname	任意模拟型标记名。

备注

此点域只能用于整型或实型标记名。

数据类型

离散型

有效值

1（真）或 0（假）

实例

下面的语句可用于“模拟输出”链接中，用于确定是否设置了标记名 **MyTag** 的 Low 报警限：

```
MyTag.LoSet;
```

{如果未设置，此处返回 0；如果已设置，则返回 1。}

参见

.Alarm, .AlarmValue, .Ack, .LoStatus, .LoLimit, .AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled, .AlarmLoEnabled, .AlarmLoInhibitor

.LoLoSet 报警

取决于模拟型标记名是否设置了 LoLo 报警限，返回值 0（假）或 1（真）。

用法 *Tagname*.**LoLoSet**

参数	描述
Tagname	任意模拟型标记名。

备注 此点域只能用于整型或实型标记名。

数据类型 离散型

有效值 1（真）或 0（假）

实例 下面的语句可用于“模拟输出”链接中，用于确定是否设置了标记名 **MyTag** 的 LoLo 报警限：

```
MyTag.LoLoSet;
```

{如果未设置，此处返回 0；如果已设置，则返回 1。}

参见 **.Alarm, .AlarmValue, .Ack, .LoLoStatus, .LoLoLimit, .AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled, .AlarmLoLoEnabled, .AlarmLoLoInhibitor**

.LoLoStatus 报警

确认是否存在 LoLo 限报警。

用法 *Tagname*.**LoLoStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 此只读点域通常等于 0。当指定标记名存在 LoLo 值报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

数据类型 离散型（只读）

有效值 0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例 当标记名 **MyTag** 的 **.LoLoStatus**（LoLo 报警限）等于 1 时，将执行下面的 IF-THEN 语句。

```
IF (MyTag.LoLoStatus == 1) THEN
   OperatorMessage="MyTag has gone into LoLo Alarm";
ENDIF;
```

参见 **.Alarm, .AlarmValue, .Ack, .LoLoLimit, .LoLoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled, .AlarmLoLoEnabled, .AlarmLoLoInhibitor**

.LoStatus 报警

确定是否存在 Low 限报警。

用法

Tagname.**LoStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

此只读点域通常等于 0。当指定标记名存在 Low 报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

数据类型

离散型（只读）

有效值

0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例

```
IF (WaterLevelTank1.LoStatus == 1) THEN
   PumpShutdown = 1;
   WaterFillValue = 1;
ENDIF;
```

参见

.Alarm, .AlarmValue, .Ack, .LoLimit, .LoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled, .AlarmLoEnabled, .AlarmLoInhibitor

.MajorDevPct 报警

监控报警检查的主偏差百分比。

用法 *Tagname*.**MajorDevPct**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画或脚本对此点域所做的更改。

数据类型 实型（读/写）

有效值 0 到 100%

实例 下面的语句将标记名 **MyTag** 的主偏差报警限属性设置为 25%：

```
MyTag.MajorDevPct=25;
```

参见 **.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevSet, .MajorDevStatus**

.MajorDevSet 报警

取决于模拟型标记名是否设置了**主偏差**百分比，返回值 0（假）或 1（真）。

用法

`Tagname.MajorDevSet`

参数	描述
Tagname	任意模拟型标记名。

备注

此点域只能用于整型或实型标记名。

数据类型

离散型

有效值

1（真）或 0（假）

实例

下面的语句可用于“模拟输出”链接中，用于决定是否设置了标记名 **MyTag** 的主偏差百分比报警限：

```
MyTag.MajorDevSet;
```

{**如果未设置，此处返回 0；如果已设置，则返回 1。**}

参见

.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevPct, .MajorDevStatus

.MajorDevStatus 报警

决定指定标记名是否存在主偏差报警。

用法 *Tagname*.**MajorDevStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 此只读点域通常等于 0。当指定标记名存在报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。

此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

数据类型 离散型（只读）

有效值 0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例 当标记名 **MyTag** 的 **.MajorDevStatus**（主偏差报警）等于 1 时，将执行下面的 IF-THEN 语句。

```
IF (MyTag.MajorDevStatus == 1) THEN
   OperatorMessage="MyTag has gone into a Major Deviation
   Alarm";

ENDIF;
```

参见 **.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevPct, .MajorDevSet**

.MaxEU 标记名

指定标记名的最大值（以工程单位表示）。

用法

`Tagname.MaxEU`

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

当服务器的值首次被 WindowViewer 接收时，它将被视为“原始值”。这些原始值可能需要放大或缩小。**.MinEU** 和 .**MaxEU** 即是用于放大或缩小原始值。作为一个典型的例子，假定域中的程序化逻辑控制器（PLC）正在读取水平尺。水平传送器送回范围在 4 到 20 mA 之间的信号。域中的 PLC 将信号转换成 0 到 4096 之间的整型值（通过数字到模拟的转换器）。此值随值被读入标记名 **TankTwoLevel**。

显示此原始值（0 到 4096 之间）不能向操作员提供有用的数据。因此有必要将该值放大或缩小到适当的工程范围。为实现这一点，必须正确设置“最小工程单位”和“最大工程单位”域。在本实例中，如果原始值 0 (4mA) 转换成“0 加仑”，而值 4096 (20mA) 转换成“100 加仑”，需要通过下面的设置在屏幕上显示正确的值：

```
Minimum Raw=0, Maximum Raw= 4096
```

```
Minimum Engineering Units=0, Maximum Engineering Units=100
```

通过这些设置，当域中的原始值为 4096 时，屏幕上显示的值将为 100。

数据类型

实型标记名和整型标记名（只读）。

有效值

取决于指定的标记名类型。

实例

由于此点域是只读点域，因此不能赋值。显示 **.MinEU** 和 **.MaxEU** 或将其用于计算很有用处，并且通过在屏幕上显示计算基础，能够使操作员加强理解。

```
DialogValueEntry ("IO_Point_717", IO_Point_717.MinEU,
IO_Point_717.MaxEU, "Please Enter a New Value:");
```

参见

.EngUnits, .MinEU, .MaxRange, .MinRange, .MinRaw, .MaxRaw, .RawValue

.MaxRange 历史

表示当前绘制趋势的标记名所显示的标记名工程单位范围百分比。

用法 *Tagname*.**MaxRange**

参数	描述
Tagname	任意历史趋势标记名。

备注 由于许多不同标记可以在任一时刻显示在历史趋势上，并且每个标记名通常具有完全不同的工程范围，所以无法以工程单位指定 **.MinRange** 和 **.MaxRange**。因此，最大和最小值以每个标记名的工程范围百分比来表示。这样，无论标记名的真正工程范围如何变化，历史趋势将只显示该标记名特定工程范围的指定*百分比*。

数据类型 实型（读/写）

有效值 **.MaxRange** 和 **.MinRange** 的限制值从 0 到 100。**.MinRange** 和 **.MaxRange** 以百分数表示。**.MinRange** 和 **.MaxRange** 以百分数表示，**.MinRange** 必须小于 **.MaxRange**。如果任一点域指定了小于 **0** 或大于 **100** 的值，这些值将定标为 **0** 或 **100**。如果 **.MinRange** 大小或等于 **.MaxRange**，趋势将不显示任何数据。

实例 下面的语句将与历史趋势标记名 **MyHistTrendTag** 关联的历史趋势的最大范围百分数设置为 25%。

```
MyHistTrendTag.MaxRange=25;
```

参见 **.ChartStart, .ChartLength, .DisplayMode, .EngUnits, .MinEU, .MaxEU, .MinRange, .MinRaw, .MaxRaw, .RawValue**

.MaxRaw 标记名

作为客户端的 WindowViewer 从 I/O 服务器收到的实际原始值的高嵌位设置。**.MaxRaw** 点域值来自于指定 I/O 标记名的标记名字典中的最大原始值域。任何超出该设置的原始值将嵌定在此值。

用法

Tagname.**MaxRaw**

参数	描述
Tagname	任意 I/O 离散、间接离散、I/O 整型、内存实型、间接模拟、I/O 消息或间接消息型标记名。

备注

此只读点域用于显示最大原始值的高嵌位设置。

数据类型

实型或整型（只读）

有效值

任何模拟值。

实例

下面的语句用于确定标记名是否超出正常操作范围，并且其值是否被嵌位。

```
IF ((Temp01.RawValue > Temp01.MaxRaw) OR (Temp01.RawValue <
   Temp01.MinRaw))THEN
   Show "Instrument Failure Window";
ENDIF;
```

参见

.EngUnits, .MinEU, .MaxEU, .MaxRange, .MinRange, .MinRaw, .RawValue

.MinEU 标记名

指定标记名的最小值（以工程单位表示）。

用法 *Tagname*.**MinEU**

参数	描述
Tagname	任意整型和实型标记名、或间接模拟标记名。

备注 当服务器的值首次被 WindowViewer 接收时，它将被视为“原始值”。这些原始值可能需要放大或缩小。**.MinEU** 和 **.MaxEU** 即是用于放大或缩小原始值。作为一个典型的例子，假定域中的程序化逻辑控制器（PLC）正在读取水平尺。水平传送器送回范围在 4 到 20 mA 之间的信号。域中的 PLC 将信号转换成 0 到 4096 之间的整型值（通过数字到模拟的转换器）。此值随值被读入标记名 **TankTwoLevel**。

显示此原始值（0 到 4096 之间）不能向操作员提供有用的数据。因此有必要将该值放大或缩小到适当的工程范围。为实现这一点，必须正确设置“最小工程单位”和“最大工程单位”域。在本实例中，如果原始值 0 (4mA) 转换成“0 加仑”，而值 4096 (20mA) 转换成“100 加仑”，需要通过下面的设置在屏幕上显示正确的值：

```
Minimum Raw=0, Maximum Raw = 4096

Minimum Engineering Units=0, Maximum Engineering Units=100
```

通过这些设置，当域中的原始值为 4096 时，屏幕上显示的值将为 100。

数据类型 实型标记名为实型及整型标记名为整型（只读）

有效值 取决于指定的标记名类型。

实例

```
AbsoluteTagRange = (Tag.MaxEU - Tag.MinEU);
```

参见 **.EngUnits, .MaxEU, .MaxRange, .MinRange, .MinRaw, .MaxRaw, .RawValue**

.MinorDevPct 报警

监控报警检查的副偏差百分比。

用法

`Tagname.MinorDevPct`

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

在标记名字典中选择“**保留参数**”选项，可以自动保存通过 WindowViewer 动画或脚本对此点域所做的更改。

数据类型

实型（读/写）

有效值

0 到 100%

实例

下面的语句将标记名 **MyTag** 的副偏差报警限属性设置为 25%：

```
MyTag.MinorDevPct=25;
```

参见

.AckDev, .AlarmDev, .AlarmMinDevDisabled, .AlarmMinDevEnabled, .AlarmMinDevInhibitor, .MinorDevSet, .MinorDevStatus

.MinorDevSet 报警

取决于模拟型标记名是否设置了副偏差百分比，返回值 0（假）或 1（真）。

用法 *Tagname*.**MinorDevSet**

参数	描述
Tagname	任意模拟型标记名。

备注 此点域只能用于整型或实型标记名。

数据类型 离散型

有效值 1（真）或 0（假）

实例 下面的语句可用于“模拟输出”链接中，用于决定是否设置了标记名 **MyTag** 的副偏差百分比报警限：

```
MyTag.MinorDevSet;
```

{**如果未设置，此处返回 0；如果已设置，则返回 1。**}

参见 **.AckDev, .AlarmDev, .AlarmMinDevDisabled, .AlarmMinDevEnabled, .AlarmMinDevInhibitor, .MinorDevPct, .MinorDevStatus**

.MinorDevStatus 报警

确定指定的标记名是否存在副偏差报警。

用法 *Tagname*.**MinorDevStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 此只读点域通常等于 0。当指定标记名存在报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。

数据类型 离散型（只读）

有效值 0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例 当标记名 **MyTag** 的 **.MinorDevStatus**（副偏差报警）等于 1 时，将执行下面的 IF-THEN 语句。

```
IF (MyTag.MinorDevStatus == 1) THEN
   OperatorMessage="MyTag has gone into a Minor Deviation
   Alarm";
ENDIF;
```

备注 此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

参见 **.AckDev, .AlarmDev, .AlarmMinDevDisabled, .AlarmMinDevEnabled, .AlarmMinDevInhibitor, .MinorDevPct**

.MinRange 历史

表示当前绘制趋势的标记名所显示的历史趋势范围百分比。

用法 *Tagname*.**MinRange**

参数	描述
Tagname	任意历史趋势标记名。

备注 由于许多不同标记可以在任一时刻显示在历史趋势上，并且每个标记名通常具有完全不同的工程范围，所以无法以工程单位指定 **.MinRange** 和 **.MaxRange**。因此，最大和最小值以每个标记名的工程范围百分比来表示。这样，无论标记名的真正工程范围如何变化，历史趋势将只显示该标记名特定工程范围的指定*百分比*。

数据类型 实型（读/写）

有效值 **.MaxRange** 和 **.MinRange** 的限制值从 0 到 100。**.MinRange** 和 **.MaxRange** 以百分数表示。**.MinRange** 和 **.MaxRange** 以百分数表示，**.MinRange** 必须小于 **.MaxRange**。如果任一点域指定了小于 **0** 或大于 **100** 的值，这些值将定标为 **0** 或 **100**。如果 **.MinRange** 大小或等于 **.MaxRange**，趋势将不显示任何数据。

实例

```
TotalChartHeight=MyHistTrendTag.MaxRange - 
MyHistTrendTag.MinRange;
```

参见 **.ChartStart, .ChartLength, .DisplayMode, .EngUnits, .MinEU, .MaxEU, .MaxRange, .MinRaw, .MaxRaw, .RawValue**

.MinRaw 标记名

作为客户端的 WindowViewer 从 I/O 服务器收到的实际原始值的低嵌位设置。.MinRaw 点域值来自于指定 I/O 标记名的标记名字典中的**最小原始**值域。任何低于该设置的原始值将嵌定在此值。

用法 *Tagname*.**MinRaw**

参数	描述
Tagname	任意 I/O 离散、间接离散、I/O 整型、内存实型、间接模拟、I/O 消息或间接消息型标记名。

备注 此只读点域用于显示最小原始值的高嵌位设置。

数据类型 实型或整型（只读）

有效值 任何模拟值。

实例 下面的语句用于确定标记名是否超出正常操作范围，并且其值是否被嵌位。

```
IF ((Temp01.RawValue > Temp01.MaxRaw) OR (Temp01.RawValue <
   Temp01.MinRaw))THEN     IF ((Temp01.RawValue >
   Temp01.MaxRaw) OR (Temp01.RawValue < Temp01.MinRaw))THEN

Show "Instrument Failure Window";

ENDIF;
```

参见 **.EngUnits, .MinEU, .MaxEU, .MaxRange, .MinRange, .MaxRaw, .RawValue**

.Name 标记名

包含“字符串”形式的指定标记名的名称。

用法 *Tagname*.**Name**

参数	描述
Tagname	任意标记名类型。

数据类型 消息型（读/写）

有效值 只能对包括报警组和 TagID 的间接型标记名的 **.Name** 域赋值。当对该点域赋值时，赋值字符串的格式必须遵从标记名命名规则，例如什么是适当的符号以及第一个字符必须是字母。

备注 此点域对下列标记名类型为只读：历史趋势、任意内存型和 I/O 型。

此点域对下列标记名类型为读/写：Tag ID 和任意间接类型。

在只读操作中，此点域不必用引号就可以指定标记名，例如，**MyTag.Name** 与 **MyTag** 相同。这对于选用 WindowMaker“**特别**”菜单上的“**更新使用计数**”命令或用交叉扫描标记名的用法来说非常有用，如果在引号内，标记名是看不到的。所以，如果标记名的唯一用法是在引号内指定，并且您执行“**更新使用计数**”，然后选择“**特别**”、“**删除无用标记**”，该标记名将被删除，因为它会被视为无用标记名。使用 "" 可清除间接标记名的引用，从而使以前的赋值无效。

在读/写操作中，此点域更为有用。它允许您修改指定间接标记名的标识。当间接标记名的 .Name 域被赋给另一标记名的名称时，假定两个标记名为同一类型，该间接标记名将实际上变成其它标记名，并且可用于 InTouch 中可使用原始标记名的任何地方。

实例

下面的只读语句使用 **.Name** 串接方法来为操作员生成消息：

```
MyMessageTag="The tagnamenamed "+ MyTag.Name+ "has a comment
of "+ MyTag.Comment";
```

方案：您开发一个图形窗口来显示有关油井的三个重要统计数据，但您有 100 口油井。可使用下面的数据改变 QuickScript 来更新单个图形窗口而不管操作员选择哪口油井。数据改变 QuickScript 的标记名为 **OilWellNumber**。

```
IndOilWellPump.Name = "OilWellPump" + Text(OilWellNumber,
"#");

IndOilWellTEP.Name = "OilWellTemp" + Text(OilWellNumber,
"#");

IndOilWellPressure.Name = "OilWellPressure" +
Text(OilWellNumber, "#");
```

此时，可用 **OilwellNumber** 作为其标记名，使用一个包含“触动输入模拟”链接的按钮来强制执行 QuickScript。在运行时，每当操作员单击按钮并输入油井号码，就可执行 QuickScript。

.NewIndex 窗口控件

返回通过 **wcAddItem( )** 或 **wcInsertItem( )** 添加到列表框或组合框的最后一个项目的对应整型索引（标记名）。

用法 [ErrorNumber=]GetPropertyI("*ControlName*.**NewIndex**",*Tagname*);

参数	描述
ControlName	窗口控件名，例如，*ListBox_4*。
Tagname	包含添加到列表框或组合框的最后一个项目的整型索引的标记名。对于空白列表，返回值 -1。

备注 此属性只在运行时可用。

数据类型 整型（只读）

应用于 列表框和组合框。

实例 下面的语句检索列表框“Listbox_1”中最近添加项目的索引，并将该值写入内存整型标记名 **NewItemIndex** 中。

```
GetPropertyI("ListBox_1.NewIndex",NewItemIndex);
```

参见 **GetPropertyI(), wcAddItem(), wcInsertItem(), .ListIndex, .TopIndex**

.NextPage 报警

当此属性从 1 变为 0 时，报警显示对象向下滚动一页（充满报警的整个屏幕）。

用法

```
[ErrorNumber=]GetPropertyD("ObjectName.NextPage",Tagname);
```

```
[ErrorNumber=]SetPropertyD("ObjectName.NextPage",Value);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数作为 "Tagname" 或 **Tagname.Name** 运行时，用于保存属性值的离散型标记名的名称。
值	一个离散值，或当函数执行时用于保存所写入值的离散型标记名。

备注

当此值从 1 变为 0 时，报警显示对象将显示下一页。一旦下一页显示完毕，此变量将自动设为 1，除非到达列表顶部，在这种情况下，此值保持为 0。

数据类型

离散型（读/写）

参见

GetPropertyD(), SetPropertyD(), .PrevPage, .PageNum, .TotalPages

.Normal 报警

当指定的标记名没有报警时，**.Normal** 等于 1。

用法

Tagname.**Normal**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟型标记名。

备注

此点域用在 QuickScript 或显示链接中，指示指定的标记名是否处于“正常”状态（无活动报警）。如果指定的标记名有一个或多个活动报警，此点域值将置为 0。

数据类型

离散型（只读）

有效值

0 = 对指定标记名有一个或多个活性报警

1 = 对指定标记名没有活性报警(缺省值)

实例

当标记名 **Tag1** 没有关联的报警时，将执行下面的 IF-THEN 语句。如果 **Tag1** 有一个或更多活动报警，则会执行“ELSE”主体：

```
IF (Tag1.Normal==1) THEN
   MyOperatorMessage="Tag1 is OK - No alarms associated
   with it";

ELSE
   MyOperatorMessage="Tag1 has one or more alarms active!";

ENDIF;
```

参见

.Alarm, .AlarmDev, .AlarmROC, .AlarmValue

.NumAlarms 报警

包含分布式报警对象内的报警数。

用法

```
[ErrorNumber=]GetPropertyI("ObjectName.NumAlarms",Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含指定的分布式报警显示对象中当前已注册的报警数目，其中不仅包括已显示的报警，也包括已注册的报警。

数据类型

整型（只读）

实例

下面的语句将分布式报警对象“AlmObj_1”所使用的当前报警数目返回给整型标记名 **AlarmCount**：

```
GetPropertyI("AlmObj_1.NumAlarms",AlarmCount);
```

参见

GetPropertyI()

.OffMsg 标记名

.OffMsg 域允许用户访问标记名字典中指定的离散型标记名的关闭消息。

用法 *Tagname*.**OffMsg**

参数	描述
Tagname	任意离散型标记名。

数据类型 消息型（读/写）。写入此域的值不会保留。

有效值 包括 0 到 15 个字符的任意字符串。

实例 如果 **MyDiscrete** 的值为 0，下面的语句可将标记名字典中 **MyDiscrete** 的 **OffMsg** 字符串赋值给离散型标记名 **StateMessage**。

```
StateMessage=Dtext (MyDiscrete, MyDiscrete.OnMsg,
MyDiscrete.OffMsg);
```

参见 **.OnMsg**

.OnMsg 标记名

.OffMsg 点域允许用户访问标记名字典中指定的离散型标记名的打开消息。

用法 *Tagname*.**OnMsg**

参数	描述
Tagname	任意离散型标记名。

数据类型 消息型（读/写）。写入此域的值不会保留。

有效值 包括 0 到 15 个字符的任意字符串。

实例

```
IF IndAnalog.OnMsg == "Running" THEN
   TypeOfTag = "IndAnalog was assigned a Motor Starter";
ENDIF
```

参见 **.OffMsg**

.PageNum 报警

包含报警对象中显示的当前页码。

用法 [ErrorNumber=]GetPropertyI("*ObjectName*.**PageNum**",*Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注 此只读点域包含指定的分布式报警显示对象中当前显示的页码。

数据类型 整型（只读）

实例 下面的语句将分布式报警对象“AlmObj_1”的当前页码返回给整型标记名 **AlarmPage**：

```
GetPropertyI("AlmObj_1.PageNum",AlarmPage);
```

参见 **GetPropertyI(), .NextPage, .PrevPage, .TotalPages**

.Pen1-.Pen8 历史

控制当前绘制历史趋势的每只笔的标记名。

用法

```
Tagname{.Pen1 | .Pen2 | .Pen3 | .Pen4 | .Pen5 | .Pen6
| .Pen7 | .Pen8};
```

参数	**描述**
Tagname	任意历史趋势标记名。

备注

考虑到使用 **.PenX** 点域的复杂性，建议尽可能使用 **HTSetPenName** 和 **HTGetPenName** 函数。

注意：只有本地标记名可赋值给 **.Pen*X*** 点域。不能使用 **provider.tag** 表示法。**provider.tag** 只能用于 **HTSetPenName**。

在学习这些点域的工作原理的一个较好方法是将历史趋势向导置于屏幕中并将其分解。这可以显示点域的用途。

数据类型

TagID（读/写）

有效值

此点域为 **.TagID** 型。这意味着可以对此点域赋予标记名的句柄。您不能赋予标记名名称给此点域，而必须赋予标记名的关联 **.TagID** 给此点域。您可以参考 **.TagID** 点域的说明来更清楚地理解这一点。一般说来，TagID 型标记名只能等价于另一 TagID 型标记名。它不能与其它标记名类型混用，除非其它标记名添加了 .TagID 扩展名。

虽然此点域被视为读/写点域，但其值不能直接显示在屏幕上。

实例

下面的语句用于将一个新标记名赋予历史趋势类型标记名 **MyHistTrendTag** 的关联历史趋势的 Pen1。通过执行下面与 **MyHistTrendTag** 关联的历史趋势脚本 Pen1 中的语句，可以开始显示标记名 **MyLoggedTag** 的历史记录数据。**MyLoggedTag** 的名称必须附加 **.TagID** 点域，以便赋值给 Pen1。由于 **.Pen1** 是 TagID 类型必须在执行赋值时匹配类型（即 **DiscreteTag = DiscreteTag** 是允许的，但 **DiscreteTag = MessageTag** 是非法的），因此必须对该点域，赋予 TagID 型对象。

```
MyHistTrendTag.Pen1=MyLoggedTag.TagID;
```

使用上面的相同例子，如果我们要显示刚赋给标记名 **MyHistTrendTag.Pen1** 的名称，情况会怎么样呢？这对操作员来说是非常有用的信息。此外，在历史趋势附近的图例中显示标记名作为参考也很有用处。为了实现这一点，您可能会想到在“消息输出”链接中显示 **MyHistTrendTag.Pen1** 的值，但是当您试着这么做时，您将看到它不起作用。为什么呢？.Pen1 点域的实际值是一个表示 WindowViewer 中的内存位置的整数，它对于显示并不特别有用。为解决这一问题，我们必须做一下练习。首先，创建一个新的 TagID 型标记名 **Pen01**，将下列语句放入上一个实例中的语句下方：

```
Pen01=MyHistTrendTag.Pen1;
```

在第一个实例中，我们将标记名 **MyLoggedTag** 赋给 **MyHistTrendTag** 的 **Pen1**。在本例中，我们进一步将 **MyHistTrendTag** 的 **Pen1** 值（现在是 **MyLoggedTag** 的 TagID）赋给标记名 **Pen01**。为什么这样做？原因是我们需要一个解决 **.Name** 的方法。我们不能简单地在屏幕上显示 **MyHistTrendTag.Name**，因为它总是显示 **MyHistTrendTag**。如果我们试图在屏幕上显示 **MyHistTrendTag.Pen1**，链接编辑器将不允许我们这样做，因为“**Pen1**”是 TagID 类型，它无法显示。因此我们需要使用一个辅助标记名来沟通显示系统和历史系统。通过将 **MyHistTrendTag.Pen1** 的值赋给 TagID 型标记名“**Pen01**”，我们实现上将 **MyLoggedTag** 赋给了“**Pen01**”。事实上，我们可以通过类似上面的方法，使用下面的一行代码来实现。它们在功能上是一致的。

```
Pen01=MyLoggedTag.TagID;
```

为什么选择第一种语句而不是第二种呢？因为第一种更通用。此外，可以方便地将 **MyHistTrendTag.Pen1** 作为数据更改脚本的“标记名”而将 **Pen01 = MyHistTrendTag.Pen1** 放入主体中。这样，无需知道哪个标记名刚赋予 **Pen1**，您可以赋予同一个标记名给 **Pen01**。这允许您经常显示赋予屏幕上历史趋势的 **Pen1** 的标记名名称，即使有人通过正常用户接口（单击 VIEW 中的趋势）或通过系统中的任意脚本改变了标记名。

进一步看此实例，有时您可能需要显示当前所绘趋势的特定标记名的最小和最大工程单位。历史趋势向导在这方面可以提供很好的帮助。我们建议您花一些时间，在屏幕上显示历史趋势向导，仔细分解并观察其组成与结构。这可以使您很好地了解这些点域的工作原理。

因为象 **Pen01** 这样的 TagID 型标记名没有工程单位，也没有通常可供标记名使用的任何其它点域，所以我们不能用它来显示赋予它的标记名工程单位。例如，我们不能显示 **Pen01.MaxEU**，因为链接编辑器会认为 .MaxEU 不是 TagID 类型的 **Pen01** 的有效点域。

为解决这个问题，我们只需使用一个辅助标记名来处理 TagID 类型和显示系统之间的对话。首先，创建一个间接型模拟标记名，如 **IndirectAnalogPen1**。在类似例 2 的数据改变脚本中，添加以下语句：

```
IndirectAnalogPen1.Name=Pen01.Name;
```

将 **Pen01** 的名称（在此实例中为 **MyLoggedTag**）赋予间接标记名 **IndirectAnalogPen1**。完成后，我们可以访问所有常用的、与模拟标记名关联的点域，在本例中是名为 **MyLoggedTag** 的整型标记名。因此，我们现在可以把模拟显示链接和下面的语句放在屏幕上：

```
IndirectAnalogPen1.MaxEU
```

这将显示当前赋予标记名 **IndirectAnalogPen1** 的标记名的最大工程单位。因为我们在上面的数据改变脚本中对其赋予 **Pen01.Name**，所以我们可以确信它会显示当前赋给标记名 **MyHistTrendTag** 的 **Pen1** 的标记名的最大工程单位。

参见 **.TagID, HTGetPenName( ), HTSetPenName( )**

.PendingUpdates 报警

指明分布式报警显示对象是否有任何等待更新的内容。

用法

[ErrorMessage=]GetPropertyI(“*ObjectName*.**PendingUpdates**”, *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如，*AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含有关指定分布式报警显示对象的等待更新信息。任何大于零的值表示分布式报警对象具有新的报警数据。此值会在每次重绘对象时重置。

数据类型

整型（只读）

实例

下面的语句返回：离散型标记名 **AlmDispStat** 的分布式报警显示对象 **AlmObj_1** 是否存在任何待定更新。

```
GetPropertyI(“AlmObj_1.PendingUpdates”,AlarmPendingUpdates);
```

参见

GetProperty

.PrevPage 报警

当此属性从 1 变为 0 时，报警显示对象向前滚动一页（充满报警的整个屏幕）。

用法

```
[ErrorNumber=]GetPropertyD("ObjectName.PrevPage",Tagname);
```

```
[ErrorNumber=]SetPropertyD("ObjectName.PrevPage",Value);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的标记名（返回相同类型）。
Value	一个离散值，或当函数执行时用于保存所写入值的离散型标记名。

备注

当此值从 1 变为 0 时，分布式报警显示对象将显示上一页。一旦上一页显示完毕，此变量将自动设为 1，除非到达列表顶部，在这种情况下，此值保持为 0。

数据类型

离散型（读/写）

参见

GetPropertyD(), SetPropertyD(), .NextPage, .PageNum, .TotalPages

.PriFrom 报警

包含当前查询所使用的低优先级。

用法

```
[ErrorNumber=]GetPropertyI("ObjectName.PriFrom",Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含指定分布式报警显示对象用于查询报警的最小优先级值。

数据类型

整型（只读）

实例

下面的语句将分布式报警对象“**AlmObj_1**”所使用的最小优先级值返回给整型标记名 **MinPri**：

```
GetPropertyI("AlmObj_1.PriFrom",MinPri);
```

参见

GetPropertyI(), .PriTo, .AlarmPri

.PriTo 报警

包含当前查询所使用的高优先级。

用法 `[ErrorNumber=]GetPropertyI("ObjectName.PriTo",Tagname);`

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注 此只读点域包含指定分布式报警显示对象用于查询报警的最大优先级值。

数据类型 整型（只读）

实例 下面的语句将分布式报警对象“**AlmObj_1**”所使用的最大优先级值返回给整型标记名 **MaxPri**：

```
GetPropertyI("AlmObj_1.PriTo",MaxPri);
```

参见 **GetPropertyI(), .PriFrom, .AlarmPri**

.ProviderReq 报警

包含当前查询所需的报警供应器数目。

用法

[ErrorNumber=]GetPropertyI("*ObjectName*.**ProviderReq**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含指定分布式报警显示对象所使用的当前查询所需的报警供应器数目。

数据类型

整型（只读）

实例

下面的语句返回分布式报警对象“**AlmObj_1**”所使用的当前查询所需的报警供应器数目，此值将写入整型标记名 **TotalProv**。

```
GetPropertyI("AlmObj_1.ProviderReq",TotalProv);
```

参见

GetPropertyI(), .ProviderRet

.ProviderRet 报警

包含已成功返回查询结果的报警供应器数目。

用法

[ErrorNumber=]GetPropertyI("*ObjectName*.**ProviderRet**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含指定分布式报警显示对象所使用的当前查询返回的报警供应器数目。

数据类型

整型（只读）

实例

下面的语句返回那些已成功向分布式报警对象“**AlmObj_1**”返回报警的报警供应器数目，此值将写入整型标记名 **RetProv**。

```
GetPropertyI("AlmObj_1.ProviderRet",RetProv);
```

参见

GetPropertyI(), .ProviderReq

.Quality 标记名

为完全理解 Wonderware 所用的质量点域，这里给出了质量标准的简短定义。Wonderware 数据质量标准基于过程控制 OLE (OPC) 的建议质量，而该质量又基于域总线数据质量规格。

质量标帜代表某个项目数据值的质量状态。该设计可能使服务器和客户端应用程序轻易地确定其所要实施的功能数量。

质量标帜的低 8 位（最不重要字节）当前以三位域的格式定义；质量、次状态和极限状态排列如下：

QQSSSSLL

质量域允许用户访问由 I/O 服务器提供的 I/O 标记名的质量。

注意：如果 I/O 连接有效，质量点域自动复位为初始值零。**.ReferenceComplete** 标志也将同时设置为零。

用法 *Tagname*.**Quality**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 值可以是 0 到 255

实例

```
IF I0Tag.Quality <> 192 THEN
   LogMessage("This data is not Good!");
ENDIF;
```

Wonderware I/O 服务器可报告送回其客户端的数据质量的六 (6) 个互异状态。它们是：

质量状态	十六进位值	OPC 质量位 MSByte xxxxxxxx	 LSByte QQSSSSLL	OPC 质量字节 .QualityStatus	.QualitySubStatus	.QualityLimit
1. 好	0x00C0	00000000	00000000	Q=3	S=0	L=0
2. 上嵌位	0x0056	00000000	00000000	Q=1	S=5	L=2
3. 下嵌位	0x0055	00000000	00000000	Q=1	S=5	L=1
4. 不能转换	0x0040	00000000	00000000	Q=1	S=0	L=0
5. 不能访问点	0x0004	00000000	00000000	Q=0	S=1	L=0
6. 通讯失败	0x0018	00000000	00000000	Q=0	S=6	L=0

当客户端应用程序无法与服务器通讯时，QualityStatus 为 0。

	0x0000	00000000	00000000	Q=0	S=0	L=0

每个质量状态的条件将报告如下：

1. 好

通讯链接已确认。
PLC 收到轮询请求并返回有效反馈包。
如果执行写入，写入过程中没有出现错误。
反馈包中的包含的数据没有转换问题。

实例 由于寄存器轮询包含 10（十进制）而返回 0x0000A 值。

2. 上嵌位

通讯链接已确认。
PLC 收到轮询请求并返回有效反馈包。
读或写寄存器没有错误。
因为值大于最大允许值，有必要将所需值限制在极限值之内。
如果是字符串，该字符串将被截短。

实例 无符号的 16 位整数被限制为 65535。

3. 下嵌位

通讯链接已确认。
PLC 收到轮询请求并返回有效反馈包。
读或写寄存器没有错误。
因为值小于最小允许值，有必要将所需值限制在极限值之内。

实例 无符号的 16 位整数被限制为 0。

4. 不能转换 通讯链接已确认。
PLC 收到轮询请求并返回有效反馈包。
来自 PLC 的数据不能转换成所需的格式。
不能转换的可能原因包括但不限于：
服务器可能返回常数而不是数据或仅返回质量信息。
数据不可用。
不知道该值是过大或过小。
从 PLC 返回的数据类型不正确。
返回浮点数，但不是数值（例如：不是一个数字）。

实例 PLC 中的 BCD 寄存器返回值 0x000A。

5. 不能访问点 通讯链接已确认。
PLC 收到轮询请求并返回有效反馈包。
PLC 报告它不能访问请求点。
无法进行访问的可能原因包括但不限于：
此项不在 PLC 内存中。
此项当前不可用（由于资源争用而锁定）。
项目不是正确的格式/数据类型。
试图写，但项目为只读。
在大多数情况下，当某个项目无效时，一组项目均将受影响。这是由服务器所使用的冻结轮询模式造成的。例如，如果在第 10 区块中的某个项目失效，那么整块会被 PLC 标记为无效。服务器会将此块中的所有项目报告为无效质量。
此数据不可用。

实例 试图读取 R40001，但 R40001 未在 PLC 的内存映像中定义。

6. 通讯失败 下列因素的任意组合：
数据通讯失败。
主题为慢轮询（或等价）模式。
没有链接验证消息。
服务器缺乏资源。例如，TSR（或驱动器）无法分配内存。
通讯链接缺乏资源。
通讯链接离线。
所有通讯通道均被占用。
网络无法将消息路由给 PLC。

实例 试图从已断电的 PLC 中读取数据。

参见 **.QualityLimit, .QualityStatus, .QualitySubstatus**

.QualityLimit 标记名

当 I/O 连接有效时，用于显示 I/O 服务器所提供的 I/O 值的质量限的整数。

用法 *Tagname*.**QualityLimit**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 (LL)

0	没有限制
1	低限
2	高限
3	常数

参见 **.Quality**

.QualityLimitString 标记名

当 I/O 连接有效时，用于显示 I/O 服务器所提供的 I/O 值的质量限字符串。

用法 *Tagname*.**QualityLimitString**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 消息型（只读）

参见 **.QualityLimit, .Quality**

.QualityStatus 标记名

当 I/O 连接有效时，用于显示 I/O 服务器所提供的 I/O 值的质量状态的整数。

用法 *Tagname*.**QualityStatus**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟间接消息型标记名。

备注 次状态位域 (QQ) 取决于质量域 (QQSSSSLL) 的值。

数据类型 整型（只读）

有效值 (QQ)

0	坏
1	不确定
3	好

参见 **.QualitySubStatus, .Quality**

.QualityStatusString 标记名

当 I/O 连接有效时，用于显示 I/O 服务器所提供的 I/O 值的质量状态字符串。

用法 *Tagname*.**QualityStatusString**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟间接消息型标记名。

数据类型 消息型（只读）

参见 **.QualityStatus, .QualitySubStatus, .Quality**

.QualitySubstatus 标记名

当 I/O 连接有效时，用于显示 I/O 服务器所提供的 I/O 值的质量次状态的整数。

用法

Tagname.**QualitySubstatus**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型

整型（只读）

有效值

(SSSS) 和 (QQ)

坏质量(QQ=0) 的次状态(SSSS):

0	非具体
1	配置错误
2	未连接
3	设备失败
4	传感器失败
5	最后已知值
6	Com 失败
7	停止服务

不确定质量 (QQ=1) 的次状态 (SSSS)：

0	非具体
1	最后可用值
4	传感器不精确
5	超出工程单位
6	次正常

好质量 (QQ=2) 的次状态 (SSSS)：

0	非具体
6	本地重写

参见

.QualityStatus, .Quality

.QualitySubstatusString 标记名

当 I/O 连接有效时，用于显示 I/O 服务器所提供的 I/O 值的质量次状态字符串。

用法

Tagname.**QualitySubstatusString**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型

消息型（只读）

参见

.QualityStatus, .QualitySubstatus, .Quality

.QueryState 报警

表示当前报警状态查询过滤器。

用法

[ErrorNumber=]GetPropertyI("*ObjectName*.**QueryState**", *Tagname*);

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含由指定分布式报警显示对象所使用的当前报警查询过滤器。

数据类型

整型（只读）

有效值

0 = 全部

1 = 未确认

2 = 确认

实例

下面的语句将分布式报警对象“AlmObj_1”的当前查询过滤器返回给整型标记名 **AlmQueryState**：

```
GetPropertyI("AlmObj_1.QueryState",AlmQueryState);
```

参见

GetPropertyI(), .QueryType

.QueryType 报警

表示当前报警查询类型。

用法

```
[ErrorNumber=]GetPropertyI("ObjectName.QueryType",Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含由指定分布式报警显示对象所使用的当前报警查询类型。

数据类型

整型（只读）

有效值

1 = 历史

2 = 摘要

实例

下面的语句将分布式报警对象“AlmObj_1”的当前查询类型返回给整型标记名 **AlmQueryType**：

```
GetPropertyI("AlmObj_1.QueryType",AlmQueryType);
```

参见

GetPropertyI(), .QueryState

.RawValue 标记名

作为客户端的 WindowViewer 从 I/O 服务器接收实际值。“原始值”域允许用户在 InTouch 缩放前访问 I/O 标记名的值。

用法 *Tagname*.**RawValue**

参数	描述
Tagname	任意 I/O 离散、间接离散、I/O 整型、内存实型、间接模拟、I/O 消息或间接消息型标记名。

备注 只读点域，用于显示在 InTouch 应用缩放前，实际的离散或模拟 I/O 值。

数据类型 任意数据类型。例如，实型标记名、离散标记名等（只读）。

有效值 可指定离散或模拟值。

实例 下面的语句可用于确定标记名是否超出正常操作范围。

```
IF ((IOtag.RawValue > IOtag.MaxRaw) OR (IOTag.RawValue <
   IOTag.MinRaw))THEN
   AlarmMessage = "Sensor is out of calibration or requires
   replacement.";

ENDIF;
```

参见 **.EngUnits, .MinEU, .MaxEU, .MinRaw, .MaxRaw**

.ReadOnly 窗口控件

确定文本框内容是只读还是读/写。

用法

`[ErrorNumber=]GetPropertyD("`*`ControlName`*`.`**`ReadOnly`**`",`
`Tagname``);`

参数	描述
ControlName	窗口控件名，例如，*Textbox_1*。
Tagname	当函数运行时用于保存属性值的离散型标记名。

备注

此属性可在开发时和运行时使用。

数据类型

离散型（只读）

有效值

0 = 文本框内容为读/写

1 = 文本框内容为只读

应用于

文本框。

实例

下面的语句检索文本框 "**TextBox_1**" 的只读状态：

```
GetPropertyD("TextBox_1.ReadOnly",A_Tagname);
```

参见

GetPropertyD(), SetPropertyD()

.Reference 标记名

允许操作员在运行期间动态改变访问名和（或）项目名。

用法 *Tagname*.**Reference**

参数	描述
Tagname	赋给 I/O 标记名的任意 I/O 型标记名或间接标记名。

备注 此点域提供一种简便的方法来动态改变标记名的访问名和（或）项目名。

数据类型 消息型（读/写）

有效值 包括访问名和（或）项目名的任意字符串。

实例 下面的语句将 I/O 整型的**项目**域设置为由文本函数生成的项目名，并由内存整型值确定。用在条件脚本中，根据 **.ReferenceComplete** 的标记序列循环：

```
MyIOTag.Reference="R" + Text( MyIndex, "#" );
{ 如果 MyIndex=40001，则结果的 ITEM 将为：R40001 }
ExcelTag.Reference="R"+Text(RowNum,"#")+"C"+Text
(ColNum, "#");
```

.ReferenceComplete 标记名

当 **.Reference** 项目标记名改变时返回确认、接收请求的项目、并在该标记名的 **.Value** 域中反映更新。

用法

Tagname.**ReferenceComplete;**

参数	描述
Tagname	赋给 I/O 标记名的任意 I/O 型标记名或间接标记名。

备注

此点域对于使用 **.Reference** 点域非常有用。**.ReferenceComplete** 指示项目和（或）访问名已经改变，更重要的是，已经从新数据源中收到一个新值。即使新数据源的值与先前数据源相同，当新数据源出现第一次更新时，系统会将此点域设为 1。

在实际更新过程中，从 **.Reference** 改变到新数据源第一次更新时间之间，系统将此点域值置为 0。

数据类型

离散型（只读）

.ROCPct 报警

监控报警检查的变化率。

用法

Tagname.**ROCPct**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注

此点域用百分比表示，它直接对应于标记名字典的报警区域内所配置的相同域。用户可以显示此点域或在脚本中使用它。此外，可以在运行时更改点域值以使更改符合当前的监视过程。

数据类型

整型（读/写）

有效值

0 到 100 %

实例

下面的语句将标记名 **MyTag** 的**变化率**百分比属性值设为 25%：

```
MyTag.ROCPct=25;
```

参见

.ROCStatus, .ROCSet

.ROCSet 报警

取决于在标记名字典中是否为模拟型标记名设置了**变化率**属性，返回值 0（假）或 1（真）。

用法

`Tagname.ROCSet`

参数	描述
Tagname	任意模拟型标记名。

备注

此点域只能用于整型或实型标记名。

Data Type

离散型

有效值

1（真）或 0（假）

实例

下面的语句可用于“模拟输出”链接中，用于决定是否设置了标记名 **MyTag** 的变化率报警限：

```
MyTag.ROCSet;
```

{**如果未设置，此处返回 0；如果已设置，则返回 1。**}

参见

.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiHiLimit, .HiLimit, .HiSet, .LoSet, .LoLoSet, .HiStatus, .HiHiStatus, .ROCPct, .ROCStatus

.ROCStatus 报警

确定指定的标记名是否存在变化率报警。

用法 *Tagname*.**ROCStatus**

参数	描述
Tagname	任意整型或实型标记名、或间接模拟标记名。

备注 此只读点域通常等于 0。当指定标记名存在变化率报警条件时，系统将其值设为 1。在报警条件消失以前，此值将一直等于 1。

数据类型 离散型（只读）

有效值 0 = 指定的报警条件不存在

1 = 指定的报警条件存在

实例 当标记名 **MyTag** 的 **.ROCStatus**（变化率报警）等于 1 时，将执行下面的 IF-THEN 语句。

```
IF (MyTag.ROCStatus == 1) THEN
   OperatorMessage="MyTag has gone into a Rate-Of-Change-
   Alarm";
ENDIF;
```

备注 此点域通常与 **.Alarm** 和 **.Ack** 点域结合使用，以确定系统内特定标记名的报警状态的确切性质。

参见 **.ROCPct, .ROCSet**

.ScooterLockLeft 历史

此点域设为 1（真）将不允许右指示器移到（超过）左指示器位置的左边。

用法 *Tagname*.**ScooterLockLeft**

参数	描述
Tagname	任意历史趋势标记名。

备注 一般说来，禁止操作员将右指示器移动到左指示器当前位置的左边是有好处的。当左指示器未锁定时，每次右指示器移到左指示器的左边，右指示器会迫使左指示器位置等于右指示器位置。

数据类型 离散型（读/写）

有效值 0 = 假 = 右指示器可以移到左指示器位置的左边

1 = 真 = 右指示器不能移到左指示器位置的左边

实例 当执行下面的语句时，与历史趋势标记名 **MyHistTrendTag** 关联的右指示器将不能移到左指示器当前位置的左边。

```
MyHistTrendTag.ScooterLockLeft=1;
```

参见 **.ScooterPosRight, .ScooterPosLeft, .ScooterLockRight**

.ScooterLockRight 历史

此点域设为 1（真）将不允许左指示器移到（超过）右指示器位置的右边。

用法 `Tagname.ScooterLockRight`

参数	描述
Tagname	任意历史趋势标记名。

备注 一般说来，禁止操作员将左指示器移动到右指示器当前位置的右边是有好处的。当右指示器未锁定时，如果左指示器移到右指示器的右边 ，左指示器会迫使右指示器位置等于左指示器位置。

数据类型 离散型（读/写）

有效值 0 = 假 = 左指示器可以移到右指示器位置的右边

1 = 真 = 左指示器不能移到右指示器位置的右边

实例 当执行下面的语句时，与历史趋势标记名 **MyHistTrendTag** 关联的左指示器将不能移到右指示器当前位置的右边。

```
MyHistTrendTag.ScooterLockRight=1;
```

参见 **.ScooterPosRight, .ScooterPosLeft, .ScooterLockLeft**

.ScooterPosLeft 历史

监控左指示器的位置。

用法

Tagname.**ScooterPosLeft**

参数	描述
Tagname	任意历史趋势标记名。

备注

此读/写点域动态控制左指示器的位置。您可以在 QuickScript 函数中使用这个点域来取得左指示器的当前位置，或者可以对此点域赋值以使左指示器位置调整到趋势上的另一位置。

此点域通常与 **HTGetValue()** 函数集结合使用。必须通知这些函数当前查询的是哪个历史趋势，以及趋势指示器的当前位置。

数据类型

实型（读/写）

有效值

0.0 到 1.0，其中 0.0 是历史趋势图表的最左端，1.0 是历史趋势图表最右端。

实例

当执行下面的语句时，将设置左指示器的新位置。在与历史趋势标记名 **MyHistTrendTag** 关联的历史趋势图表中，左指示器会移到与图表最左端距离图表宽度 34% 的位置。

```
MyHistTrendTag.ScooterPosLeft=.34;
```

在下面的语句中，使用 QuickScript 函数 **HTGetValueAtScooter()** 来取得左指示器当前位置的 Pen1 值。由于函数的参数列表中任何变量的改变都会造成函数重新求值，所以每次左指示器的位置改变时，此语句将重新求值。

```
MyRealTag=HTGetValueAtScooter
(MyHistTrendTag,MyHistTrendTag.UpdateCount,1,
MyHistTrendTag.ScooterPosLeft,1,"PenValue");
```

参见

.ScooterPosRight, .ScooterLockLeft, .ScooterLockRight

.ScooterPosRight 历史

监控右指示器的位置。

用法

Tagname.**ScooterPosRight**

参数	描述
Tagname	任意历史趋势标记名。

备注

此读/写点域动态控制右指示器的位置。您可以在 QuickScript 函数中使用这个点域来取得右指示器的当前位置，或者可以对此点域赋值以使右指示器位置调整到趋势上的另一位置。

此点域通常与 **HTGetValue()** 函数集结合使用。必须通知这些函数当前查询的是哪个历史趋势，以及趋势指示器的当前位置。参见下面的例子。

数据类型

实型（读/写）

有效值

0.0 到 1.0，其中 0.0 是历史趋势图表的最右边，1.0 是历史趋势图表最左边。

实例

当执行下面的语句时，将设置右指示器的新位置。在与历史趋势标记名 **MyHistTrendTag** 关联的历史趋势图表中，右指示器会移到与图表最右端距离图表宽度 34% 的位置。

```
MyHistTrendTag.ScooterPosRight=.34;
```

下面的语句使用 QuickScript 函数 **HTGetValueAtScooter()** 来取得右指示器当前位置的 **Pen1** 值。由于函数的参数列表中任何变量的改变都会造成函数重新求值，所以每次右指示器的位置改变时，此语句将重新求值。

```
MyRealTag=HTGetValueAtScooter
(MyHistTrendTag,MyHistTrendTag.UpdateCount,2,
MyHistTrendTag.ScooterPosRight,1,"PenValue");
```

参见

.ScooterPosLeft, .ScooterLockLeft, .ScooterLockRight

.Successful 报警

显示当前查询是否成功。

用法

`[ErrorNumber=]GetPropertyD("`*`ObjectName`*`.`**`Successful`**`",`
`Tagname``);`

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时用于保存属性值的离散型标记名。

备注

此只读点域包含指定分布式报警显示对象所用的最后一次查询的状态。

数据类型

离散型（只读）

有效值

0 = 查询错误

1 = 查询成功

实例

下面的语句将分布式显示对象“**AlmObj_1**”的最后一次查询的状态返回给离散标记名 **AlmFlag**：

```
GetPropertyD("AlmObj_1.Successful",AlmFlag);
```

参见

GetPropertyD()

.SuppressRetain 报警

读/写分布式报警显示对象的保持抑制功能的状态。

用法

`[ErrorNumber=]GetPropertyD("`*`ObjectName`*`.`**`SuppressRetain`**`",` *`Tagname`*`");`

`[ErrorNumber=]SetPropertyD("`*`ObjectName`*`.`**`SuppressRetain`**`",` *`Tagname`*`");`

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时用于保存属性值的离散型标记名。

数据类型

离散型（读/写）

有效值

0 = 关闭保持

1 = 开启保持

实例

下面的语句通过离散型标记名 **SupRtn** 设置 "**AlmObj_1**" 的抑制保持器的状态：

```
SetPropertyD("AlmObj_1.SuppressRetain", SupRtn);
```

参见

GetPropertyD(), SetProperty()

.TagID 标记名

与历史趋势 **.Pen1-.Pen8** TagID 标记名结合使用，以监控趋势笔当前绘制趋势的标记名。

用法 `Tagname.TagID`

参数	描述
Tagname	任意离散、整型或实型标记名、间接离散或间接模拟型标记名。

备注 **.TagID** 提供标记名句柄，主要用在给历史趋势笔赋予标记名的环境中。

数据类型 TagID（只读）

实例

```
MyHistTrendTag.Pen6=SomeAnalogTag.TagID;
```

参见 **.Pen1-.Pen8**

.TimeDate 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示自 I/O 服务器提供 I/O 值以来已过的整数天数。

用法 *Tagname*.**TimeDate**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

参见 **.TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear**

.TimeDateString 标记名

以 Windows 注册表中设置的格式显示日期的字符串。

用法 `Tagname`**.TimeDateString**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 消息型（只读）

参见 **.TimeDate, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear**

.TimeDateTime 标记名

实型**标记名点域**，当 I/O 连接有效时，用于显示自 I/O 服务器提供 I/O 值以来已过的小数天数。

用法 *Tagname*.**TimeDateTime**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 实型（只读）

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear**

.TimeDay 标记名

整型标记名点域，当 I/O 链接有效时，用来显示 I/O 服务器提供 I/O 值的那一天。

用法 *Tagname*.**TimeDay**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 1 到 31。

参见 **.TimeDate, .TimeDateString, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear**

.TimeHour 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示 I/O 服务器提供 I/O 值的小时值。

用法 *Tagname*.**TimeHour**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 0 到 23。

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear**

.TimeMinute 标记名

整型 Tagname.field，当 I/O 链接有效时，用来显示 I/O 服务器提供 I/O 值时的分钟。

用法

`Tagname.TimeMinute`

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型

整型（只读）

有效值

0 到 59。

参见

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeMonth 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示 I/O 服务器提供 I/O 值的月份。

用法 *Tagname*.**TimeMonth**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 1 到 12。

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear**

.TimeMsec 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示 I/O 服务器提供 I/O 值的时间（以毫秒计）。

用法

Tagname.**TimeMsec**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型

整型（只读）

有效值

0 到 999。

参见

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeSecond 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示 I/O 服务器提供 I/O 值的时间（以秒计）。

用法 *Tagname*.**TimeSecond**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 0 到 59。

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeTime, .TimeTimeString, .TimeYear**

.TimeTime 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示自午夜以来 I/O 服务器提供 I/O 值的时间（以毫秒计）。

用法 `*Tagname*.TimeTime`

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 0 到 86399999。

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTimeString, .TimeYear**

.TimeTimeString 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示 I/O 服务器提供 I/O 值的时间值。

用法 *Tagname*.**TimeTimeString**

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 消息型（只读）

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeYear**

.TimeYear 标记名

整型**标记名点域**，当 I/O 连接有效时，用于显示 I/O 服务器提供 I/O 值的四位年份。

用法 `Tagname.TimeTime`

参数	描述
Tagname	任意离散、整型或实型标记名、或间接模拟或间接消息型标记名。

数据类型 整型（只读）

有效值 以 #### 格式表示的任意年号，例如 1998。

参见 **.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString**

.TopIndex 窗口控件

确定列表框中最顶端项目的相应整型索引。

用法

[ErrorNumber=]GetPropertyI("*ControlName*.**TopIndex**", *Tagname*);

[ErrorNumber=]SetPropertyI("*ControlName*.**TopIndex**", *Number*);

参数	描述
ControlName	窗口控件名，例如，*ListBox_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。
Number	定义列表框中最顶端项目的索引号。

备注

此属性只在运行时可用。

数据类型

整型（读/写）

应用于

列表框。

实例

下面的语句将列表框对象“Listbox_1”的最顶端索引设为 14：

```
SetPropertyI("ListBox_1.TopIndex",14);
```

参见

GetPropertyI(), SetPropertyI(), .ListIndex, .NewIndex

.TotalPages 报警

包含分布式报警对象的总的页数（充满报警的整个屏幕）。

用法

```
[ErrorNumber=]GetPropertyI("ObjectName.TotalPages",
Tagname);
```

参数	描述
ObjectName	分布式报警对象的名称。例如 *AlmObj_1*。
Tagname	当函数运行时，用于保存属性值的整型标记名。

备注

此只读点域包含指定分布式报警显示对象中包含的总的报警页数。

数据类型

整型（只读）

实例

下面的语句将分布式报警对象“**AlmObj_1**”中包含的报警总页数返回给整型标记名 **AlmTotalPage**：

```
GetPropertyI("AlmObj_1.TotalPages",AlmTotalPages);
```

参见

GetPropertyI(), .NextPage, .PrevPage, .PageNum

.UnAck 报警

控制本地报警的报警确认状态。

用法

Tagname.UnAck**=0**

参数	**描述**
Tagname	任意离散、整型或实型标记名、或间接模拟型标记名或报警组。

备注

此点域设为 0 可确认与指定标记名/组关联的任何未确认报警。当指定的标记名为报警组时，指定组中与标记名关联的所有未确认报警均将得到确认。如果指定的标记名是任意其它类型，则只有与该标记名关联的未确认离散报警才会得到确认。将此点域设为 0 以外的值没有意义，其结果未定义。

数据类型

离散型（只读/重置）

有效值

0

实例

下面的语句确认与标记名“**Tag1**”关联的任意报警。

```
Tag1.UnAck=0;
```

下面的语句确认报警组 **PumpStation** 内的所有未确认报警。

```
PumpStation.UnAck = 0;
```

注意：.UnAck 有一个逆向点域叫 **.Ack**。当报警已确认后，**.Ack** 置为 **1**。

参见

.Ack, Ack(), .Alarm, .AlarmAckModel

.UpdateCount 历史

每次关联趋势出现更新时递增。

用法 *Tagname*.**UpdateCount**

参数	描述
Tagname	任意历史趋势标记名。

备注 此点域直接链接到历史检索子系统。当请求与指定历史趋势标记关联的历史趋势的新数据时，历史检索子系统转到磁盘检索指定数据。一旦完成检索过程，此点域会递增 1，**.UpdateCount** 在与历史数据趋势有关的许多函数调用中用作触发器来重新计算函数。

数据类型 整型（只读）

有效值 任何正整数

实例 下面的语句使用 QuickScript 函数 **HTGetValueAtScooter()** 来取得右指示器当前位置的 **Pen1** 值。由于函数参数列表中任何变量的改变都会造成函数重新求值，因此每次检索（更新）完成后，**.UpdateCount** 的值将递增，此语句将重新将求值。

```
MyRealTag=HTGetValueAtScooter
(MyHistTrendTag,MyHistTrendTag.UpdateCount,2,
MyHistTrendTag.ScooterPosRight,1,"PenValue");
```

参见 **.UpdateInProgress, .UpdateTrend**

.UpdateInProgress 历史

如果历史检索正在进行当中，值等于 1；否则为 0。

用法 *Tagname*.**UpdateInProgress**

参数	描述
Tagname	任意历史趋势标记名。

备注 此点域直接链接到历史检索子系统。当请求与指定历史趋势标记关联的历史趋势的新数据时，历史检索子系统转到磁盘检索指定数据。在检索过程中此点域设为 1，一旦此过程完成，**.UpdateInProgress** 重置为 0，**.UpdateInProgress** 用在许多与历史数据趋势有关的函数调用中。

大多数历史趋势屏幕有一种机制，通过它操作员能在趋势数据中“滚动”。当操作员操纵屏幕控制时，历史子系统会确认屏幕上的数据是否为当前数据。如果操作员滚动趋势到一个不是当前显示或存在于内存中的一个区域，子系统将转到磁盘检索所请求的数据。由于此过程需要一段时间来完成，系统为历史趋势屏幕的设计者提供了一种方法，可以通知操作员所请求的数据正在检索当中。利用这一反馈，操作员可以知道系统正在执行所请求的任务。

数据类型 离散型（只读）

有效值 0 = 没有更新在进行中

1 = 更新正在进行

实例 下面的语句一般用作历史趋势“滚动”按钮之上或其附近的文本对象上的“可见性”链接中的表达式。当历史子系统正在检索请求的数据时，此表达式值为 1，否则，如果趋势完成更新，此表达式值将置为 0。

```
MyHistTrendTag.UpdateInProgress
```

参见 **.UpdateCount, .UpdateTrend**

.UpdateTrend 历史

导致历史趋势图表用所有当前值更新。

用法 `Tagname.UpdateTrend`

参数	描述
Tagname	任意历史趋势标记名。

备注 历史趋势不会自动自我更新。要更新图表并显示指定标记名的当前值，必须更改图表开始值或图表长度等。通过在触动按钮动作 QuickScript 中使用此点域，操作员可以在运行时随时更新图表。

如果与历史趋势关联的其它点域将要发生更改，您也可以在 QuickScript 中使用此点域。这可以确保历史趋势图表上显示最新的数据。

将此点域设为 1 以外的值没有意义，其结果未定义。

数据类型 离散型（只写）

有效值 1

实例 下面的语句导致与历史趋势标记名 **MyHistTrendTag** 关联的历史趋势使用所有参数的当前值进行更新。

```
MyHistTrendTag.UpdateTrend=1;
```

.Value 标记名

包含指定标记名的值。这是系统内每个 InTouch 标记名的缺省点域。 如果未指定其它点域，则假定使用此点域。

用法

Tagname.**Value**

参数	描述
Tagname	除历史趋势标记名之外的任意标记名。

备注

这是系统内每个 InTouch 标记名的缺省点域。 如果未指定其它点域，则假定使用此点域。此点域很少用到，但在某些情况下，为使计算或参数用途更清楚，可将其用作文档化工具。

数据类型

取决于(相同于)指定标记名类型(读/写)。

实例

以下语句设置名为“**MyTag**”的内存整型标记的值为 100：

```
Tagname.Value=100;
```

它在功能上等价于：

```
Tagname=100;
```

.Value 窗口控件

所有 InTouch 窗口控件向导的缺省属性。此属性更改在 InTouch 标记名和窗口控件向导中保持同步。

用法

```
[ErrorNumber=]GetPropertyM("ControlName[.Value]",Tagname);
[ErrorNumber=]SetPropertyM("ControlName[.Value]", Value);
[ErrorNumber=]GetPropertyI("ControlName[.Value]",Tagname);
[ErrorNumber=]SetPropertyI("ControlName[.Value]",Value);
[ErrorNumber=]GetPropertyD("ControlName[.Value]",Tagname);
[ErrorNumber=]SetPropertyD("ControlName[.Value]",Value);
```

注意：赋予列表框或组合框的标记名的初始值不能用于初始化列表框或组合框的值。

参数	描述
ControlName	窗口控制的名称，如 *ChkBox_4*。
Tagname	包含列表内的项目整数计数的有效标记名。
[**.Value**]	此属性是可选的。如果未指定，函数总是假定使用 **.Value** 属性。
Value	要写入的实际值，或者当函数执行时，用于保存要写入的属性值的有效 InTouch 标记名（与要写入的属性类型相同）。

备注

此属性在开发和运行时均为读/写。如果 **.Value** 通过标记名与列表框或组合框的关联来访问，则为只读。如果 **.Value** 被赋予复选框，单选按钮或文本框等，它为可读/写的。开发时指定的值将充当运行时的缺省值。

数据类型

文本框、列表框和组合框为消息型（读/写）。

单选钮为整型（读/写）。

复选框为离散型（读/写）。

应用于

文本框、列表框、组合框、复选框和单选钮。

实例

下面的语句将单选钮对象“RadioButton_1”的值设为 4：

```
SetPropertyI( "RadioButton_1.Value",4 );
```

参见

GetPropertyM(), SetPropertyM(), GetPropertyI(), SetPropertyI(), GetPropertyD(), SetPropertyD()

.Visible 窗口控件

决定窗口控件在窗口中是否可见。

用法

```
[ErrorNumber=]GetPropertyD("ControlName.Visible",Tagname);
[ErrorNumber=]SetPropertyD("ControlName.Visible",Number);
```

参数	描述
ControlName	窗口控件名，例如，*ListBox_1*。
Tagname	当函数运行时，用于保存属性值的标记名（返回相同类型）。
Number	一个离散值，或当函数执行时用于保存所写入值的离散型标记名。

备注

该属性在开发和运行时均为读/写。

数据类型

离散型（读/写）

有效值

0 = 控件不可见

1 = 控件可见

应用于

文本框、列表框、组合框、复选框和单选钮。

实例

下面的语句创建一个名为“**TextBox_1**”的不可见文本框：

```
SetPropertyD("TextBox_1.Visible",0);
```

参见

GetPropertyD(), SetPropertyD()

第 3 章

InTouch QuickScript 函数

InTouch QuickScript 是 InTouch 应用程序最强大的功能之一。InTouch QuickScript 功能允许在指定标准满足时执行特定的命令和逻辑操作。例如，按下键、打开窗口，改变值等。

QuickFunction 是从其它 QuickScript 和动画链接表达式调用的 QuickScript（参见下面的附注）。系统将可重复使用的编码存储在单个 QuickScript 和单个位置中，从而支持在一个编辑进程中更新所有 QuickScript 实例。

注意：您必须使用“触发器”标记名作为 QuickFunction 的参数，以强制更新动画链接。例如，使用标记名 $Second 作为 QuickFunction 的参数可以在每次 $Second 改变值时对动画链接表达式求值，从而每隔一秒调用 QuickFunction 一次。

通过使用 InTouch QuickScript，您可以创建各种自定义和自动化系统函数。

Abs() 数学

返回指定数字的绝对值（无符号的等价值）。

语法

`Result=`**`Abs`**`(`*`Number`*`);`

参数	描述
Number	任意数字、实型或整型标记名。

备注

计算 *Number* 的绝对值并返回给 *Result*。

实例

Abs(14) 返回 14

Abs(-7.5) 返回 7.5

Ack() 报警

确认任何未确认的 InTouch 报警。

语法

`Ack` `Tagname;`

参数	描述
Tagname	任意 InTouch 标记名、报警组或组变量。

备注

此函数可应用于标记名、报警组或组变量（组变量是被赋予报警组名称的标记名）。

实例

下面的语句可在按钮上用于确认任何未确认的报警：

```
Ack $System; (All alarms)
Ack Tagname;
Ack GroupName;
Ack GroupVariable;
```

参见

almAckAll(), almAckGroup() almAckTag(), almAckDisplay(), almAckRecent(), almAckPriority(). almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

ActivateApp() 系统

激活另一个当前正在运行的 Windows 应用程序。

语法

`ActivateApp` *`TaskName`*;

参数	描述
TaskName	此函数将激活的任务。

备注

TaskName 是出现在任务栏或任务管理器中的准确文本字符串（包括空格），可以在 Windows NT 中通过右击任务栏，然后单击任务管理器或 Ctrl+Alt+Delete 来访问。

实例

下面的实例检查命令提示符是否正在运行。如果正在运行，把它放到前台并作为屏幕中心。否则，出现命令提示符并从命令提示符启动 DOS 程序 edit.com。

```
IF InfoAppActive( InfoAppTitle("cmd")) == 1 THEN
   ActivateApp InfoAppTitle("cmd");
ELSE
StartApp "cmd /c edit";
ENDIF;
```

参见

StartApp(), InfoAppTitle()

almAckAll() 报警

确认当前请求的所有报警，包括当前未显示在报警摘要显示对象中的报警。

语法

`[Result=]almAckAll(ObjectName,Comment);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
MessageTag = “Acknowledge All by” + $Operator;
almAckAll(“AlmObj_1”, MessageTag);
```

参见

Ack(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckDisplay() 报警

仅确认报警摘要显示对象中当前可见的那些报警。

语法

```
[Result=]almAckDisplay(ObjectName,Comment);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
almAckDisplay("AlmObj_1", "Display Acknowledgement");
```

参见

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckGroup() 报警

作为上次查询显示的结果，并且在结果报警包含相同报警组名和供应器名时，确认指定分布式报警对象实例中包含的所有报警。

语法

[Result=]almAckGroup(O*bjectName*, `ApplicationName`, *GroupName*, *Comment)*;

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
注释	报警确认注释。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
MessageTag = "Acknowledge group, Turbines, by" + $Operator;
almAckGroup("AlmObj_1", "\Intouch", "Turbine", MessageTag);
```

参见

Ack(), almAckAll(), almAckDisplay(), almAckTag(), almAckRecent(), almAckPriority(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckPriority() 报警

作为上次查询显示的结果，并且在结果报警处于同名供应器和报警组的指定优先级范围时，确认指定分布式报警对象实例中包含的所有报警。

语法

```
[Result=]almAckPriority(ObjectName, ApplicationName,
GroupName, FromPri, ToPri, Comment);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
FromPri	报警的开始优先级，例如 100。
ToPri	报警的结束优先级，例如 900。
注释	报警确认注释。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
almAckPriority(“AlmObj_1”,“\\node1\Intouch”, “Turbines”,
10, 100, “almAckPriorityComment”);
```

参见

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckRecent() 报警

确认最近发生的报警。

语法

```
[Result=]almAckRecent(ObjectName,Comment);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
almAckRecent(“AlmObj_1”, $DateString);
```

参见

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckSelect() 报警

仅确认报警摘要显示对象中选定的那些报警。

语法

`[Result=]almAckSelect(ObjectName,Comment);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
IF ($Hour > 0 and $Hour < 8) THEN
   AckTag = “NightShift”;
ELSE
   AckTag = “Day Shift”;
ENDIF;
almAckSelect (“AlmObj_1”,AckTag);
```

参见

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckSelectedGroup() 报警

确认与指定分布式报警显示控件实例中选定的一个或多个报警具有相同组名的、包含相同供应器和组名的所有报警。

语法

`[Result=]almAckSelectedGroup(ObjectName,Comment);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注

实例

```
MessageTag = “Acknowledge selected groups by” + $Operator;
almAckSelectedGroup (“AlmObj_1”, MessageTag);
```

参见

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedPriority(), almAckSelectedTag()

almAckSelectedPriority() 报警

确认与指定分布式报警显示控件实例中选定的一个或多个报警具有相同优先级值的、包含相同供应器和组名的所有报警。

语法 [Result=]**almAckSelectedPriority**(*ObjectName, Comment)*;

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注 从选定的报警记录计算优先级，并采用最小和最大优先级。

实例

```
MessageTag = "Acknowledge selected priorities by" + $Operator;
almAckSelectedPriority ("AlmObj_1", MessageTag);
```

参见 **Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedTag()**

almAckSelectedTag() 报警

确认具有来自相同报警供应器和组名的相同标记名，并且与指定分布式报警显示控件实例中的一个或多个所选报警具有相同优先级的所有报警。

语法 [Result=]**almAckSelectedTag**(*ObjectName,Comment)*;

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Comment	报警确认注释。

备注

实例

```
MessageTag = "Acknowledge selected tagnames by" + $Operator;
almAckSelectedTag ("AlmObj_1", MessageTag);
```

参见 **Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority()**

almAckTag() 报警

作为显示上次查询及给定标记名的结果，确认指定分布式报警显示实例中包含的所有报警。

语法

```
[Result=]almAckTag(ObjectName, ApplicationName, GroupName, Tagname, FromPri, ToPri, Comment);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
Tagname	报警标记名名称。
FromPri	报警的开始优先级，例如 100。
ToPri	报警的结束优先级，例如 900。
Comment	报警确认注释。

备注

实例

```
almAckTag(“AlmObj_1”, “\\node1\Intouch”, “Turbines”, “Valve1”, 10, 100, “Value”, “LoLo”, “almAckTagComment”);
```

参见

Ack(), almAckAll(), almAckGroup(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag(0

almDefQuery() 报警

使用缺省属性执行查询以更新指定的分布式报警显示实例。

语法

```
[Result=]almDefQuery(ObjectName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

缺省查询属性在开发时指定。

有关所返回错误号的列表，请参阅附录 A。

实例

```
almDefQuery("AlmObj_1");
```

参见

almQuery(), almSetQueryByName()

almMoveWindow() 报警

滚动指定的分布式报警显示实例窗口。

语法

```
[Result=]almMoveWindow(ObjectName,Options,Repeat);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
Option	要执行的操作类型：

类型	描述
LineDn	下移一行。
LineUp	上移一行。
PageDn	下翻一页。
PageUp	上翻一页。
Top	转到列表的顶部。
Bottom	转到列表的底部。
PageRt	右移一页。
PageLf	左移一页。
Right	转到列表的结尾（右侧）。
Left	转到列表的开头（左侧）。

参数	描述
Repeat	重复此操作的次数。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
almMoveWindow("AlmObj_1", "Bottom", 0);
almMoveWindow("AlmObj_1", "LineDn", 3);
almMoveWindow("AlmObj_1", "PageUp", 0);
```

almQuery() 报警

执行查询以更新指定的分布式报警显示实例。

语法

```
[Result=]almQuery(ObjectName,AlarmList,FromPri,ToPri,State,
Type);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
AlarmList	设定报警查询/名称管理器别名，以对例如 "\intouch!$System" 或消息型标记名等执行查询。
FromPri	所显示报警的开始优先级。例如，100 或整型标记名。
ToPri	所显示报警的结束优先级。例如，900 或整型标记名。
State	指定要显示的报警类型。例如，“UnAck”或消息型标记名。有效状态为“全部 l”、“未确认”或“确认”。
Type	指定查询类型，例如“Hist”（历史报警）或“Summ”（摘要报警）。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

此语句检索在 **MyAlarmListGroup** 中指定的、优先级介于 500 和 600 之间的所有历史报警，这些报警将列出在报警显示“Alarm1”中。

```
almQuery("AlmObj_1","MyAlarmListGroup",500,600,"All",
"Hist");
```

下面是 AlarmList 参数的有效语法：

```
MyAlarmList
```

此处，**MyAlarmList** 是从“**名称管理器**”设置的报警列表。

```
\intouch!GroupA
```

此处，GroupA 是本地节点上定义的报警组。

```
\\NodeXInTouch!GroupB
```

此处，GroupB 是在节点 X 上定义的报警组。

参见

almDefQuery(), almSetQueryByName()

almSelectAll() 报警

切换选择指定分布式报警显示实例中的所有报警。

语法

[Result=]**almSelectAll**(*ObjectName*);

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
If $AccessLevel > 8000 THEN
almSelectAll("AlmObj_1");
almAckSelect("AlmObj_1", "Ack Selected by a Manager");
ENDIF;
```

参见

almSelectItem(), almSelectGroup(), almSelectPriority(), almSelectTag(), almUnSelectAll()

almSelectGroup() 报警

作为上次查询显示的结果，并且在结果报警包含相同报警组名时，切换选择指定分布式报警对象实例中包含的所有报警。

语法

[Result=]**almSelectGroup**(ObjectName, ApplicationName, GroupName);

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
almSelectGroup("AlmObj_1", "\InTouch", "Turbine");
```

参见

almSelectAll(), almSelectItem(), almSelectPriority(), almSelectTag(), almUnSelectAll()

almSelectItem() 报警

切换选择报警显示对象中辉亮显示的项目。

语法

`[Result=]almSelectItem(ObjectName);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
almSelectItem("AlmObj_1");
```

参见

almSelectAll(), almSelectGroup(), almSelectPriority(), almSelectTag(), almUnSelectAll()

almSelectPriority() 报警

作为上次查询显示的结果，并且在结果报警在指定的优先级范围内时，切换指定分布式报警显示实例中包含的所有报警。

语法

`[Result=]almSelectPriority (ObjectName, ApplicationName, GroupName, FromPri, ToPri);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
FromPri	报警的开始优先级，例如，100 或整型标记名。
ToPri	报警的结束优先级，例如，900 或整型标记名。

实例

```
almSelectPriority("AlmObj_1", "\\node1\Intouch",
"Turbines", 10, 100);
```

参见

almSelectAll(), almSelectItem(), almSelectGroup(), almSelectTag(), almUnSelectAll()

almSelectTag() 报警

作为显示上次查询及给定标记名的结果，切换指定分布式报警显示实例中包含的所有报警。

语法

[Result=]**almSelectTag** (ObjectName, ApplicationName, GroupName, Tagname, FromPri, ToPri);

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
Tagname	报警标记名名称。
FromPri	报警的开始优先级，例如，100 或整型标记名。
ToPri	报警的结束优先级，例如，900 或整型标记名。

备注

实例

almSelectTag(“AlmObj_1”, “\\node1\Intouch”, “Turbines”, “Valve1”, 10, 100);

参见

almSelectAll(), almSelectItem(), almSelectGroup(), almSelectPriority(), almUnSelectAll()

almSetQueryByName() 报警

使用与自定义（常用）查询名关联的查询参数，启动分布式报警显示控件的指定实例的新报警查询。

语法

[Result=]**almSetQueryByName**(*ObjectName, QueryName)*;

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
QueryName	使用“常用查询”创建的查询名。

备注

此查询用于分布式报警显示的特定实例。屏幕上可能有多个此类显示对象，每个显示对象均有自己的查询。

实例

almSetQueryByName(“AlmObj_1”, “Turbine Queries”);

参见

almQuery(), almDefQuery()

almShowStats() 报警

显示报警显示对象的统计屏幕。

语法

```
[Result=]almShowStats(ObjectName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注 有关所返回错误号的列表，请参阅附录 A。

实例

```
almShowStats("AlmObj_1");
```

almSuppressAll() 报警

抑制显示当前查询中的所有当前及未来报警实例，包括当前未显示在报警摘要显示对象中的实例。

语法

```
[Result=] almSuppressAll(ObjectName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注 此函数类似于 **almAckAll()**，通过确定在上下滚动查看显示对象时所看到的所有报警，来确定要抑制的报警。

实例

```
almSuppressAll("AlmObj_1");
```

参见 **almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()**

almSuppressGroup() 报警

抑制显示属于指定组名的当前及未来报警实例。

语法

```
[Result=]almSuppressGroup(ObjectName, ApplicationName,
GroupName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\InTouch。
GroupName	报警组名，例如 $System。

实例

```
almSuppressGroup("AlmObj_1", "¥InTouch", "Turbines");
```

参见

almSuppressAll(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressDisplay() 报警

抑制显示报警摘要显示对象中可见的那些报警的当前及未来实例。

语法

`[Result=]`**`almSuppressDisplay`**`(ObjectName);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

此函数类似于对应的 **almAckDisplay()** 函数，通过确定当前显示的所有报警，来确定要抑制的报警。

实例

```
almSuppressDisplay("AlmObj_1");
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressPriority() 报警

抑制具有相同报警供应器名和组名的、给定优先级范围内的任何报警的当前和未来显示。

语法

`[Result=]`**`almSuppressPriority`**`(ObjectName, ApplicationName, GroupName, FromPri, ToPri);`

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
FromPri	报警的开始优先级，例如，100 或整型标记名。
ToPri	报警的结束优先级，例如，900 或整型标记名。

实例

```
almSuppressPriority("AlmObj_1", "\\node1\Intouch",
"Turbines", 10, 100);
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressRetain() 报警

保持所有以下查询的报警抑制。

语法

```
[Result=]almSuppressRetain (ObjectName, SuppressionRetainFlag);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
SuppressionRetain Flag	任意离散或模拟型标记名、0 或非零值。如果保留下列查询的抑制信息，其值为真；否则为假。

备注

如果标帜为零，则当报警查询改变时，抑制过滤器会被删除。

实例

```
almSuppressRetain(“AlmObj_1”, TRUE);
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressSelected() 报警

抑制显示报警摘要显示对象中选定报警的未来实例。

语法

```
[Result=]almSuppressSelected (ObjectName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

此函数类似于 **almAckSelect()** 函数，通过在显示对象中选定报警来确定报警。

实例

```
almSuppressSelected(“AlmObj_1”);
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressTag() 报警

抑制具有相同报警供应器名、组名和优先级范围的、由给定标记名发出的任何报警的当前和未来显示。

语法

```
[Result=]almSuppressTag(ObjectName, ApplicationName,
GroupName, TagName, FromPri, ToPri, AlarmClass, AlarmType);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。
ApplicationName	应用程序名，例如 \\node1\Intouch。
GroupName	报警组名，例如 $System。
TagName	报警标记名名称。
FromPri	报警的开始优先级，例如，100 或整型标记名。
ToPri	报警的结束优先级，例如，900 或整型标记名。
AlarmClass	报警的报警类，例如“值报警”。
AlarmType	报警的报警类型，例如 “HiHi”。

备注

实例

```
almSuppressTag(“AlmObj_1”, “\\node1\Intouch”, “Turbines”,
“Valve1”, 10, 100, “Value”, “LoLo”);
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressSelectedGroup() 报警

抑制与指定分布式报警显示控件中的同名报警供应器内的一个或多个所选报警处于相同报警组的任何报警的当前和未来显示。

语法

[Result=]**almSuppressSelectedGroup***(ObjectName)*;

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

此函数类似于 **almAckSelectedGroup()** 函数，先确定所选报警，然后确定其所属的报警组，再抑制这些报警组中的未来的报警。

实例

```
almSuppressSelectedGroup(“AlmObj_1”);
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressSelected(), almSuppressSelectedPriority(), almSuppressSelectedTag()

almSuppressSelectedPriority() 报警

抑制与指定分布式报警显示控件中的同名报警供应器和报警组标记内的一个或多个所选报警处于相同报警优先级的任何报警的当前和未来显示。

语法

```
[Result=]almSuppressSelectedPriority(ObjectName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

从选定的报警记录计算优先级，并采用最小和最大优先级。

此函数类似于 **almAckSelectedPriority()** 函数，先确定显示对象中选定的报警，然后确定这些报警的对应优先级，再抑制这些优先级的未来报警。

实例

```
almSuppressSelectedPriority("AlmObj_1");
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTagName(), almSuppressDisplay(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedTag(), almAckSelectedPriority()

almSuppressSelectedTag() 报警

抑制与同名报警供应器内的一个或多个所选报警具有相同标记名的任何报警的当前和未来显示。

语法

```
[Result=]almSuppressSelectedTag(ObjectName);
```

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

实例

```
almSuppressSelectedTag("AlmObj_1");
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressSelectedAlarm(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almAckSelectedTag(), almUnSuppressAll()

almUnSelectAll() 报警

撤消选定在指定分布式报警显示实例中选定的所有报警。

语法

[Result=]**almUnSelectAll**(*ObjectName*);

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

备注

有关所返回错误号的列表，请参阅附录 A。

实例

```
If $AccessLevel == 9999 THEN
   almAckSelect("AlmObj_1", "Comment");{This alarm can be
   acknowledged by only Administrator}
ELSE
   almUnSelectAll("AlmObj_1");
ENDIF;
```

参见

almSelectAll(), almSelectItem(), almSelectGroup(), almSelectPriority(), almSelectTag()

almUnSuppressAll() 报警

清除所有被抑制的报警。

语法

[Result=] **almUnSupressAll***(ObjectName)*;

参数	描述
ObjectName	报警对象名，例如 AlmObj_1。

实例

```
almUnSupressAll("AlmObj_1");
```

参见

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag()

ArcCos() 数学

对于给定的介于 -1 和 1 之间（包含 -1 和 1）的数字，此函数返回一个其*余弦*等于该数字的介于 0 和 180 度之间的角度。

语法

`Result=ArcCos(Number);`

参数	描述
Number	任意数字、实型或整型标记名。

备注

计算 *Number* 的反余弦值并返回给 *Result*。

有关所返回错误号的列表，请参阅附录 A。

实例

ArcCos(1) 返回 0

ArcCos(-1) 返回 180

参见

Cos(), Sin(), Tan(), ArcSin(), ArcTan()

ArcSin() 数学

对于给定的介于 -1 和 1 之间（包含 -1 和 1）的数值，此函数返回一个其*正弦*等于该数字的介于 -90 和 90 度之间的角度。

语法

`Result=ArcSin(Number);`

参数	描述
Number	任意数字、实型或整型标记名。

备注

计算 *Number* 的正弦值并返回给 *Result*。

有关所返回错误号的列表，请参阅附录 A。

实例

ArcSin(1) 返回 90

ArcSin(-1) 返回 -90

参见

Cos(), Sin(), Tan(), ArcCos(), ArcTan()

ArcTan() 数学

对于给定的数字，此函数返回一个其正切等于该数字的介于 -90 和 90 度之间的角度。

语法

`Result=ArcTan(Number);`

参数	描述
Number	任意数字、实型或整型标记名。

备注

计算 *Number* 的反正切值并返回给 *Result*。

有关所返回错误号的列表，请参阅附录 A。

实例

ArcTan(1) 将返回 45

ArcTan(0) 将返回 0

参见

Cos(), Sin(), Tan(), ArcCos(), ArcSin()

ChangePassword() 安全性

显示允许已登录的操作员改变其口令的“**改变口令**”对话框。

语法

`[Result=]ChangePassword();`

参数	描述
[Result]	返回下列整数值之一： 0 = 按下“取消” 1 = 按下“确定”

备注

如果使用触摸屏应用程序，会有一个使用字母数字键盘的选项。

实例

```
Errmsg=ChangePassword();
```

此 QuickScript 如果置于按钮上或者根据条件脚本或数据改本脚本来调用，则会打开一个对话框（带可选键盘），提示用户输入当前口令和新口令并验证新口令。

Cos() 数学

返回给定角（以度表示）的*余弦*值。

语法

```
Result=Cos(Number);
```

参数	描述
Number	任意数字、实型或整型标记名。

备注 计算 *Number* 的余弦并返回给 *Result*。

实例

Cos(90) 返回 0

Cos(0) 返回 1

```
Wave = 50 * Cos(6 * $Second);
```

参见 **Sin(), Tan(), ArcCos(), ArcSin(), ArcTan()**

DialogStringEntry() 其它

在屏幕上显示字母数字键盘，允许操作员改变标记名字典中消息型标记名的当前字符串值。

语法

```
[Result=]DialogStringEntry(MessageTag_Text,
UserPrompt_Text);
```

参数	描述
[Result]	返回下列整数值之一： 0 = 按下“取消” 1 = 按下“确定” -1 = 内部错误 -2 = 无法初始化 -3 = 未定义标记名 -4 = 标记名非消息型 -5 = 无法写入
MessageTag_Text	指定被修改的消息型标记名的名称。此函数要求该参数为一字符串，因此您必须用引号将名称括起或使用不带引号的 **.Name** 域来识别此标记名。此外也允许使用作为指针的消息型标记。参见下面的例子。
UserPrompt_Text	指定显示在键盘顶部的用户消息。

备注 此函数主要用于包含触摸屏的应用程序。

实例

```
Errmsg=DialogStringEntry(MyMessageTag.Name,  "Enter a new string...");
```

```
Errmsg=DialogStringEntry("MyMessageTag","Enter a new string...");
```

这些参数也可以充当其它标记名的“指针”，以允许函数在运行时为动态。

```
Errmsg=DialogStringEntry(MessageTagX, MessageDisplay);
```

使用上面的例子，可对每个指针标记名指定输入链接，在执行函数前允许用户修改函数参数的任意一项或全部。

例如，下面的 QuickScript 会显示一个字母数字键盘，当键盘顶端显示消息“Enter a new string…”时，允许修改 **MyMessageTag**：

```
MessageTagX="MyMessageTag"; {将字符串“MyMessageTag”（实际要修改的标记名）赋予内存消息型标记名 MessageTagX}
```

```
MessageDisplay="Enter a new string…"; {将新的消息型字符串赋予内存消息型标记名 MessageDisplay}
```

```
Errmsg=DialogStringEntry(MessageTagX, MessageDisplay); {无需引号，因为 MessageTagX 定义为消息型标记名}
```

参见 **DialogValueEntry()**

DialogValueEntry() 其它

在屏幕上显示数字小键盘，允许操作员更改标记名字典中离散型、整型或实型标记名的当前值。

语法

```
[Result=]DialogValueEntry(ValueTag_Text,
LowLimit,HighLimit,UserPrompt_Text);
```

参数	描述
[Result]	返回下列整数值之一： 0 = 按下“取消” 1 = 按下“确定” -1 = 上限<=下限 -2 = 无法初始化 -3 = 未定义标记名 -4 = 标记名非离散、整型或实型 -5 = 写入失败
ValueTag_Text	指定被修改的离散、整型或实型标记名的名称（此函数要求该参数为一字符串，因此您必须用引号将名称括起或使用不带引号的 **.Name** 域，或使用消息型标记名作为指针来识别此标记名。参见下面的例子）。
LowLimit	指定标记名的最小允许值（应大于或等于此标记名的最小值、最小原始值或最小工程单位的定义值）。
HighLimit	指定标记名的最大允许值（应大于或等于此标记名的最小值、最小原始值或最小工程单位的定义值）。
UserPrompt_Text	指定显示在小键盘顶部的用户消息。

备注

此函数主要用于包含触摸屏的应用程序。

实例

```
Errmsg=DialogValueEntry(MyIntegerTag.Name,
MyIntegerTag.MinEU, MyIntegerTag.MaxEU, "Enter a new
value...");
```

```
Errmsg=DialogValueEntry("MyIntegerTag", -100, 100, "Enter a
new value...");
```

这些参数也可以充当指向其它标记的“指针”，允许函数在运行时为动态。

```
Errmsg=DialogValueEntry(TagnameX, Min, Max,
MessageDisplay);
```

使用上面的例子，可对每个指针标记名指定输入链接，在执行函数前允许用户修改函数参数的任意一项或全部。

例如，下面的 QuickScript 将显示一个数字小键盘，当键盘顶端显示消息“Enter a new value…”时，允许在下限和上限分别为¨100 到 100 的范围内修改 *MyIntegerTag*：

TagnameX="MyIntegerTag"; {将字符串“MyIntegerTag”（实际要修改的标记名）赋予内存消息型标记名 TagnameX }

Min=-100; {将标记名的最小允许值赋予内存实型/整型标记名 Min }

Max=100; {将标记的最大允许值赋予内存实型/整型标记 Max }

MessageDisplay="Enter a new value..."; {将新的消息型字符串赋予内存消息型标记名 MessageDisplay }

Errmsg=DialogValueEntry(TagnameX, Min, Max, MessageDisplay); {无需引号，因为 TagnameX 定义为消息型标记名。通过对 TagmameX 赋予离散、整型或实型标记名，此函数可以修改该指定标记名}

参见 **DialogStringEntry()**

DText() 字符串

根据离散型标记名的值动态改变消息型标记名。

语法

```
MsgTag=DText(Discrete_Tag,OnMsg,OffMsg);
```

参数	描述
MsgTag	消息型标记名。
Discrete_Tag	离散型标记名。
OnMsg	当 *Discrete_Tag* 等于 1（真、开、是）时所显示的消息。
OffMsg	当 *Discrete_Tag* 等于 0（假、关、否）时所显示的消息。

实例

```
MessageTag=Dtext(DiscreteTag, DiscreteTag.OnMsg,
DiscreteTag.OffMsg);
If Dtext(D_Tag, “True”, “False”) == “True” THEN
  Message=”D_Tag contains a value of 1”;
ELSE
  Message=”D_Tag contains a value of 0”;
ENDIF;
```

Exp() 数学

返回 e 的某次幂。

语法

```
Result=Exp(Number);
```

参数	描述
Number	任意数字、实型或整型标记名。

备注

计算 *Number* 的指数并返回给 *Result*。

实例

```
Exp(1) 返回 2.718...
```

此函数的取值范围为 -88.72 到 88.72。

FileCopy() 系统

复制 *SourceFile* 到 *DestFile*，与 DOS Copy 命令或 Windows 文件中的复制功能相似。

语法

```
[Result=]FileCopy(SourceFile,DestFile,DoneTag);
```

参数	描述
SourceFile	源文件名（包括完整路径）。
DestFile	目标文件名（包括完整路径）或目录名（参见下面的例子）。
DoneTag	**FileCopy()** 函数用于报告复制过程进度的标记名名称。此参数必须是一个指示标记名名称的字符串（而非实际标记名本身）。所以，如果完成的标记名叫做 Monitor，您应该提供“Monitor”或 *Monitor.name*，而不只是 Monitor。

备注

当使用 **FileCopy()** 函数时，如果过程成功启动，则立即返回 1。如果另一过程正在运行（此新过程无法启动），它将返回 0；如果出现错误，则返回 -1。使用此返回值，可以监视 **FileCopy()** 函数的启动：

```
Status=FileCopy("C:\*.TXT", "C:\BACKUP", "Monitor");
```

Status 是一个为 1、-1 或 0 的整型标记名。**FileCopy()** 函数在后台执行，这样就不会干扰 InTouch 的运行。*DoneTag* 允许应用程序或用户监视此复制操作的*进度*。通过这种方法，可以提醒用户在转移过程启动*后*可能发生的任何错误。这与上述返回的 *Status* 不同，在那里，*Status* 指示的是复制过程是否已成功*启动*。

一旦复制过程已成功启动，*DoneTag* 就会被赋值。如果此过程尚在运行中，将置为 0；如果此过程成功结束，则置为 1；如果或在此过程结束前发生错误，则置为 -1。

SourceFile 和 *DestFile* 一般为文件名。不过，用 **FileCopy()** 函数复制*单一文件*时，目标可以是一个目录：

```
FileCopy("C:\DATA.TXT", "C:\BACKUP", "Monitor");
```

此函数可以把文件“DATA.TXT”复制到“C:\”驱动器上一个叫做“BACKUP”的目录下。标记名 Monitor 在此完成后将置为 1。

不过，如果 *SourceFile* 包含任何通配符的话，则 *DestFile* 必须是一个目录（而非文件名），否则此函数将返回一个错误代码：

```
FileCopy("C:\*.TXT", "C:\BACKUP", "Monitor");
```

这将从 C:\ 根目录将所有的 .TXT 文件复制到 C:\BACKUP 目录下。标记名 Monitor 在此完成后将置为 1。

注意：当在异步 QuickFunction 中使用时，**FileCopy()** 无法工作。**FileCopy()** 只能用在同步 QuickFunctions 中。

参见 **FileDelete(), FileMove(), FileReadFields(), FileReadMessage(). FileWriteFields(), FileWriteMessage()**

FileDelete() 系统

删除不需要或不想要的文件。

语法

```
[Result=]FileDelete(Filename);
```

参数	描述
Filename	要删除的文件名。

备注 若找到要删除的文件，并成功地删除，此函数将返回 1，否则此函数返回 0。

注意：FileDelete() 不接受通配符。例如 * 和 ?。使用通配符会返回错误 (0) “找不到文件”。

当在异步 QuickFunction 中使用时，**FileDelete()** 无法工作。**FileDelete()** 只能用在同步 QuickFunctions 中。

实例

```
Status=FileDelete("C:\DATA.TXT");
```

若在 C:\ 下找到“DATA.TXT”文件，则 *Status* 等于 1，否则为 0。

参见 **FileCopy(), FileMove(), FileReadFields(), FileReadMessage(). FileWriteFields(), FileWriteMessage()**

FileMove() 系统

类似于 **FileCopy()** 函数，不同的是它将文件从一个位置移动到另一个位置，而不是制作一份拷贝。

语法

```
[Result=]FileMove(SourceFile,DestFile,DoneTag);
```

参数	描述
SourceFile	源文件名（包括完整路径）。
DestFile	目标文件名（包括完整路径）。
DoneTag	**FileMove()** 函数用于报告移动过程的进度的标记名名称。此参数必须是一个指示标记名名称的字符串（而非实际标记名本身）。所以，如果完成的标记名叫做 Monitor，您必须提供“Monitor”或 Monitor.name 而不只是 Monitor。

备注

当使用 **FileMove()** 函数时，如果过程成功启动，将立即返回 1。如果另一过程正在运行（此新过程无法启动），则返回 0；如果出现错误，则返回 –1。使用此返回值，可以监视 **FileMove()** 函数的启动。

```
Status=FileMove("C:\DATA.TXT", "D:\DATA.TXT", "Monitor");
```

Status 是一个为 1、-1 或 0 的整型标记名。**FileMove()** 函数在后台执行，这样就不会干扰 InTouch 的运行。使用 *DoneTag* 是为了允许应用程序或用户监视移动操作的*进度*。通过这种方法，可以提醒用户在转移过程启动*后*可能发生的任何错误。这与上述返回的 *Status* 不同，在那里，*Status* 指示的是移动过程是否已成功*启动*。

注意：当在异步 QuickFunction 中使用时，**FileMove()** 无法工作。**FileMove()** 只能用在同步 QuickFunctions 中。

一旦转移过程已成功启动，*DoneTag* 就会被赋值。如果此过程尚在运行中，将置为 0；如果此过程成功结束，则置为 1；如果或在此过程结束前发生错误，则置为 –1。

若 *SourceFile* 和 *DestFile* 位于同一驱动器上，此函数可以简单地更改此文件的目录引用（在这里计算机保持磁盘上的文件名和存储位置），而不用实际移动任何数据。在这种情况下，不管此文件的大小，移动操作可以很快完成。若 *SourceFile* 和 *DestFile* 位于不同的驱动器上，移动操作所花的时间将随文件的大小不同而不同。这是因为数据必须从一个物理磁盘传送到另一物理磁盘上。

实例

```
FileMove ("C:\DATA.TXT","C:\BACKUP\DATA.TXT","Monitor");
```

这将把文件“DATA.TXT”从“C”驱动器的根目录下复制到一个叫做“BACKUP”的目录下。标记名 Monitor 在此完成后将置为 1。

注意：当 *SourceFile* 和 *DestFile* 指定了相同目录中的不同文件名时，此函数也可用于重命名文件。

```
FileMove ("C:\DATA.TXT","C:\DATA.BAK","Monitor");
```

参见

FileCopy(), FileDelete(), FileReadFields(), FileReadMessage(). FileWriteMessage(),FileWriteFields()

FileReadFields() 系统

从指定的文件读取逗点分离变量 (CSV) 记录。

语法

```
[Result=]FileReadFields(Filename,FileOffset,StartTag,
NumberOfFields);
```

参数	描述
Filename	指定要读取的文件。
FileOffset	指定文件中开始读取的位置。
StartTag	指定写入*第*一个数据项的 InTouch 标记名的名称。此标记名名称必须以一个数字结尾（如 **MyTag1**）。此参数必须是一个指示标记名名称的字符串（而非实际标记名本身）。所以，若您的标记名称为 **MyTag 1**，您就得给出“**MyTag1**”或 **MyTag1**.name，而不仅仅是 MyTag1。
NumberOfFields	指定要读取的域数目（文件中每个记录以逗号分隔的域的数目）。

备注

若 *StartTag* 为“**MyTag1**”而 *NumberOfFields* 为 3，则有 3 个域从文件中读出并保存在 **MyTag1**、**MyTag2** 和 **MyTag3** 中。这些具有连续名称的标记名必须事先在 InTouch 中创建，并且可以属于不同的类型（整型、消息等）。

实例

```
If the first line of C:\DATA\FILE.CSV is:This is text,
3.1416, 5
```

下面的 QuickScript 将读出此行，并把“This is text”保存在 **MyTag1** 中、3.1416 保存在 **MyTag2** 中、5 保存在 **MyTag3** 中：

```
BytePosition=FileReadFields ("C:\DATA\FILE.CSV", 0,
"MyTag1", 3);
```

此函数返回读取之后的新字节位置。您可以使用此返回值作为在下次读取时的 *FileOffset* 值。

```
FileReadFields
("C:\DATA\FILE.CSV",BytePosition,"MyTag1",3);
```

注意： InTouch 中的消息型标记名最多可以存储 131 个字符。

参见

FileCopy(), FileMove(), FileDelete(), FileReadMessage(), FileWriteMessage(), FileWriteFields()

FileReadMessage() 系统

从指定的文件读取指定的字节数（或整行）。

语法

```
[Result=]FileReadMessage(Filename,FileOffset,Message_Tag,
CharsToRead);
```

参数	描述
Filename	指定要读取的文件。
FileOffset	指定文件中开始读取的位置。
Message_Tag	指定存放从文件中读取的数据的位置。最多可储存 131 个字符。
CharsToRead	指定从文件中读取的字节数。要处理文本文件，*CharsToRead* 可置为 0。此函数将读取直到文件中的下一个 LF（换行符），或总共 131 个 字符，看哪一个先到达。

实例

```
FileReadMessage ("C:\DATA\FILE.TXT", 0, MsgTag, 0);
```

从文件“C:\DATA\FILE.TXT”读取第一行并将其保存到 **MsgTag** 中。此函数返回读取之后的新字节位置。您可以使用此返回值作为在下次读取时的 *FileOffset* 值。

参见

FileCopy(), FileMove(), FileDelete(), FileReadFields(). FileWriteFields(), FileWriteMessage()

FileWriteFields() 系统

将逗号分离变量 (CSV) 记录写入指定文件。

语法

```
[Result=]FileWriteFields(Filename,FileOffset,StartTag,
NumberOfFields);
```

参数	描述
Filename	指定要写入的文件。如果文件名不存在，则会创建一个。
FileOffset	指定文件内开始写入的位置。如果 FileOffset 为 -1，函数将写入到文件结尾。
StartTag	指定包含第一个数据项的 InTouch 标记名的名称。此标记名名称必须以一个数字结尾（如 **MyTag1**）。此参数必须是一个指示标记名名称的字符串（而非实际标记名本身）。所以，若您的标记名称为 **MyTag 1**，您就得给出“**MyTag1**”或 **MyTag1**.name，而不仅仅是 MyTag1。
NumberOfFields	指定要写的域数目（文件中每条记录以逗号分隔的域的数目）。

备注

若 *StartTag* 为“**MyTag1**”，而 *NumberOfFields* 为 3，则有 3 个域将写入文件中（从 **MyTag1**、**MyTag2** 和 **MyTag3**）。这些具有连续名称的标记必须事先在 InTouch 中创建，并且可以属于不同的类型（整型，消息等等）。

实例

下面的 QuickScript 将行 "This is text, 3.1416, 5" 写入 C:\DATA\FILE.CSV 的第一行。"This is text" 值当前位于 **MyTag1** 中，3.1416 位于 **MyTag2** 中，5 位于 **MyTag3** 中：

```
FileWriteFields ("C:\DATA\FILE.CSV", 0, "MyTag1", 3);
```

此函数返回写入之后的新字节位置。您可以使用此返回值作为下次写入时的 *FileOffset* 值。

下面的 QuickScript 将行 "This is text, 3.1416, 5" 写入 C:\DATA\FILE.CSV 的最后一行。"This is text" 值当前位于 **MyTag1** 中，3.1416 位于 **MyTag2** 中，5 位于 **MyTag3** 中：

```
FileWriteFields ("C:\DATA\FILE.CSV", -1, "MyTag1", 3);
```

参见

FileCopy(), FileDelete(), FileMove(), FileReadFields(), FileReadMessage().FileWriteMessage()

FileWriteMessage() 系统

将指定的字节数（或整行）写入指定的文件。

语法

[Result=]**FileWriteMessage**(Filename,FileOffset,Message_Tag,LineFeed);

参数	描述
Filename	指定要写入的文件。如果文件名不存在，则会创建一个。
FileOffset	指定文件内开始写入的位置。如果 FileOffset 为 -1，函数将写入到文件结尾。
Message_Tag	指定要写入文件的字符。
LineFeed	指定是否在写操作后添加一个换行符。当写入一个文本文件时，将 LineFeed 设为 1。

备注

此函数返回写入之后的新字节位置。您可以使用此返回值作为下次写入时的 FileOffset 值。

实例

下列的语句将一个名为 ***MsgTag*** 的消息型标记名写入 *C:\DATA\FILE.TXT* 文件的结尾：

```
FileWriteMessage ("C:\DATA\FILE.TXT", -1, MsgTag, 1);
```

参见

FileCopy(), FileDelete(), FileMove(), FileReadFields(), FileReadMessage(), FileWriteFields()

GetNodeName() 系统

返回 NetDDE 节点名给字符串变量。

语法

`GetNodeName(`*`Tagname`*`,`*`NodeNum`*`);`

参数	描述
Tagname	保存节点名的 InTouch 消息型标记名。
NodeNum	指定消息型标记名的字符长度的一个整数。

备注

当执行此 QuickScript 时，**GetNodeName()** 函数会读取本地节点名，并将其返回给 *Tagname*（*NodeNum* 设置消息型标记名的字符长度）。

实例

```
GetNodeName("MyNodeTag",131);
If MyNodeTag == “Master” THEN
  MessageTag = “This is the Primary Machine!”;
ENDIF;
```

GetPropertyD() GOT

在运行时检索指定属性的离散值。

语法

`[ErrorNumber=]`**`GetPropertyD`**`("ControlName[.Property]",Tagname);`

参数	描述
ControlName	窗口控件名，例如 *ChkBox_1*；或报警对象名，例如 AlmObj_1。
.Property	窗口控件或报警对象属性。 有关这些属性的详细信息，请参阅第 2 章“点域”。
Tagname	在函数处理时存储属性值的有效 InTouch 标记名（将返回相同类型）。

备注

有关所返回错误号的列表，请参阅附录 A。

参见

GetPropertyI(), GetPropertyM(), SetPropertyD(), SetPropertyI(), SetPropertyM()

GetPropertyI() GOT

在运行时检索指定属性的离散值。

语法

```
[ErrorNumber=]GetPropertyI("ControlName.[Property]",Tagname)
;
```

参数	描述
ControlName	窗口控件名，例如 *ChkBox_1*；或报警对象名，例如 AlmObj_1。
.Property	窗口控件或报警对象属性。 有关这些属性的详细信息，请参阅第 2 章“点域”。
标记名	在函数处理时存储属性值的有效 InTouch 标记名（将返回相同类型）。

备注 有关所返回错误号的列表，请参阅附录 A。

参见 **GetPropertyD(), GetPropertyM(), SetPropertyD(), SetPropertyI(), SetPropertyM()**

GetPropertyM() GOT

在运行时检索指定属性的消息值。

语法

```
[ErrorNumber=]GetPropertyM("ControlName.[Property]",Tagname)
;
```

参数	描述
ControlName	窗口控件名，例如 *ChkBox_1*；或报警对象名，例如 AlmObj_1。
.Property	窗口控件或报警对象属性。 有关这些属性的详细信息，请参阅第 2 章“点域”。
标记名	在函数处理时存储属性值的有效 InTouch 标记名（将返回相同类型）。

备注 有关所返回错误号的列表，请参阅附录 A。

参见 **GetPropertyD(), GetPropertyI(), SetPropertyD(), SetPropertyI(), SetPropertyM()**

Hide 其它

在 QuickScript 中隐藏各个窗口。**Hide()** 函数必须位于每个要隐藏窗口的名称之前。

语法 `Hide` *Window*;

参数	描述
Window	窗口名或一个包含窗口名的消息型标记名。

备注 在运行期间，如果窗口不存在，此语句将被忽略。如果此 QuickScript 只是隐藏或显示窗口，最好使用“触动按钮”链接来显示窗口或隐藏窗口，而不要用此函数。

注意：如果使用消息型标记名作为隐藏窗口的参数，将会出现下面的错误消息：

“需要窗口名 - 必须是字符串表达式”

为避免再次出现此消息，请在消息型标记名前面加上加号 (+)。
例如：

```
DIM Test AS message;
Test = "MyWindow1";
Hide ""+Test;
```

实例 `Hide "My Popup Alarm Window";`

参见 **Hideself, Show, ShowAt(), ShowTopLeftAt(), ShowHome**

HideSelf 其它

隐藏当前的活动窗口。

语法 `HideSelf;`

备注 此函数只能用在 Touch 触动按钮 QuickScript 中。

参见 **Hide, Show**

HTGetLastError() 历史

确定在最后一次检索指定笔时是否出现错误。

语法 `[Result=]`**`HTGetLastError`**`(Hist_Tag,UpdateCount,PenNum);`

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名名称。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
PenNum	表示笔号（从 1 到 8）的整型标记名或值。
[Result=]	此函数可能返回下列结果：

结果	描述
0	没有错误
1	一般服务器错误
2	旧的请求
3	文件错误
4	未加载服务器
5	函数内传递的趋势/笔不存在
6	数据库中不存在此趋势名标记
7	传给函数的笔号无效（不在 1 到 8 之间）
8	笔号不存在在标记名/非记录标记名。

实例 下面的语句检索最近一次用标记名 *Trend1* 检索趋势的 *Pen3* 时出现的错误，并把结果赋给整型标记名 *ResultCode*：

```
[ResultCode=]HTGetLastError("Trend1",Trend1.UpdateCount,3);
```

在动画“模拟值显示”QuickScript 中使用下面的语句：

```
HTGetLastError("Trend1",Trend1.UpdateCount,3);
```

HTGetPenName() 历史

返回指定趋势的指定笔号当前所用的标记名。

语法

```
MessageResult=HTGetPenName(Hist_Tag,UpdateCount,PenNum);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
PenNum	表示笔号（从 1 到 8）的整型标记名或值。函数返回一个表示指定笔的标记名的消息型标记名。

实例

下面的语句用标记名 **Trend1** 检索趋势笔 *Pen2* 的标记名，并将结果赋给消息型标记名 *TrendPen*：

```
TrendPen=HTGetPenName("Trend1",Trend1.UpdateCount,2);
```

HTGetTimeAtScooter() 历史

返回由 *ScootName* 和 *ScootLoc* 指定的指示器位置处的样本自 1970 年 1 月 1 日 GMT 00:00:00 以来的时间（以秒为单位）。

语法

```
IntegerResult=HTGetTimeAtScooter(Hist_Tag,UpdateCount,
ScootNum,ScootLoc);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
ScootNum	表示左或右指示器的整数。 1=左指示器 2=右指示器
ScootLoc	表示趋势的 **.ScoeterPosRigth** 或 **.ScooterPosLeft** 域的实数。

备注

当 *UpdateCount*、*ScootNum* 和 *ScootLoc* 中任一参数发生变化时，会对表达式重新求值。这可以确保在新的检索或指示器移动后对表达式重新求值。

实例

下面的语句为趋势 *Trend1* 的左指示器的当前位置值检索以秒为单位的时间值：

```
HTGetTimeAtScooter("Trend1",Trend1.UpdateCount,1,
Trend1.ScooterPosLeft);
```

HTGetTimeStringAtScooter() 历史

返回由 *ScootNum* 和 *ScootLoc* 指定的指示器位置处的样本包含时间/日期的字符串。

语法

```
MessageResult=HTGetTimeStringAtScooter(Hist_Tag,UpdateCount
,
ScootNum,ScootLoc,Format_Text);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
ScootNum	表示左或右指示器的整数。 1=左指示器 2=右指示器
ScootLoc	表示趋势的 **.ScoeterPosRigth** 或 **.ScooterPosLeft** 域的实数。
Format_Text	指定要使用的时间/日期格式的字符串。下面是允许的 *Format_Text* 字符串： "Date"、"Time"、"DateTime"、"DOWShort"（例如 Wed）和 "DOWLong"（例如 Wednesday）。

备注

当 *UpdateCount*、*ScootNum* 和 *ScootLoc* 中任一参数发生变化时，会对表达式重新求值。这可以确保在新的检索或指示器移动后对表达式重新求值。字符串的格式决定返回值的内容。

实例

下面的语句为趋势 *Trend1* 的右指示器的当前位置值检索日期/时间值。这个值保存在消息型标记名 *NewRightTimeString* 中，格式是“*Time*”：

```
NewRightTimeString=HTGetTimeStringAtScooter
("Trend1",Trend1.UpdateCount,2,Trend1.ScooterPosRight,
"Time");
```

HTGetValue() 历史

返回整个趋势的指定笔的请求类型值。UpdateCount 参数在检索完成后会对表达式求值。

语法

```
RealResult=HTGetValue(Hist_Tag,UpdateCount,PenNum,
ValType_Text);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
PenNum	表示笔号（从 1 到 8）的整型标记名或值。
ValType_Text	指示返回值类型的字符串：

类型	描述
"PenAverageValue"	整个趋势的平均值。
"PenMaxValue"	整个趋势的最大值。
"PenMinValue"	整个趋势的最小值。
"PenMaxEU"	整个趋势的最大工程单位。
"PenMinEU"	整个趋势的最小工程单位。
"PenStdDev"	整个趋势的标准偏差。

注意：在 **HTGetValue** 函数中使用 *ValType_Text* 参数时，确保使用上述有效类型之一。

备注

返回一个表示给定类型的计算值的内存实型标记名。

实例

下面的语句取得为趋势 *Trend1* 的 *Pen2* 所检索的数据的标准偏差。此值保存在内存实型标记名 *LeftHemisphereSD* 中：

```
LeftHemisphereSD=HTGetValue("Trend1",Trend1.UpdateCount,
2, "PenStdDev");
```

HTGetValueAtScooter() 历史

返回指定的指示器位置，趋势和笔号处的样本的请求类型值。使用 *UpdateCount* 参数，可在检索完成后对表达式求值。

语法

```
RealResult=HTGetValueAtScooter(Hist_Tag,UpdateCount,
ScootNum,ScootLoc,PenNum,ValType_Text);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
ScootNum	表示左或右指示器的整数。 1=左指示器 2=右指示器
ScootLoc	表示趋势的 **.ScoeterPosRigth** 或 **.ScooterPosLeft** 域的实数。
PenNum	表示笔号（从 1 到 8）的整型标记名或值。
ValType_Text	指示返回值类型的字符串：

类型	描述
"PenValue"	指示器位置的值。
"PenValid"	如果值无效，返回 0，否则返回 1。

注意：在 **HTGetValueAtScooter** 函数中使用 *ValType_Text* 参数时，确保使用上述有效类型之一。

备注

返回一个表示 "PenValue" 的计算值的内存实型标记名；返回一个表示“PenValue”的值的内存离散型标记名。

实例

如果此值是一个实际样本，下面的语句保存 1；如果此值对右指示器当前位置的内存离散型标记名 *ValidFlag* 中的趋势 *Trend1* 的笔 *Pen3* 无效，则下面的语句保存 0：

```
ValidFlag=HTGetValueAtScooter("Trend1",Trend1.UpdateCount,
2, Trend1.ScooterPosRight,3,"PenValid");
```

HTGetValueAtZone() 历史

为包含在某个趋势的指定笔的左右指示器之间的数据返回请求类型值。使用 *UpdateCount* 参数，可在检索完成后对表达式求值。

语法

```
RealResult=HTGetValueAtZone(Hist_Tag,UpdateCount,
Scoot1Loc,Scoot2Loc,PenNum,ValType_Text);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
UpdateCount	表示趋势的 **.UpdateCount** 域的整数。
Scoot1Loc	表示趋势的 **.ScooterPosLeft** 域的实数。
Scoot2Loc	表示趋势的 **.ScooterPosRight** 域的实数。
PenNum	表示笔号（从 1 到 8）的整型标记名或值。
ValType_Text	指示返回值类型的字符串：

类型	描述
"PenAverageValue"	左、右指示器之间的区域平均值。
"PenMaxValue"	左、右指示器之间区域的最大值。
"PenMinValue"	左、右指示器之间区域的最小值。
"PenMaxEU"	左、右指示器之间区域的最大工程单位。
"PenMinEU"	左、右指示器之间区域的最小工程单位。
"PenStdDev"	左、右指示器之间区域的标准偏差。

注意：在 **HTGetValueAtZone** 函数中使用 *ValType_Text* 参数时，确保使用上述有效类型之一。

备注

返回一个表示给定类型的计算值的内存实型标记名。对 *Scoot1Loc* 和 *Scoot2Loc* 指定常数值不起作用，它仅用于为更新显示行而触发函数的处理过程。此函数直接使用运行时数据库的趋势标记名的 **.ScooterPosLeft** 和 **.ScooterPosRight** 域来确定区域边界。

实例

下面的语句为趋势 *"Trend1" Pen1* 的左右指示器之间的区域数据计算平均值。所得值保存在内存实型标记名 *AvgValue* 中：

```
AvgValue=HTGetValueAtZone("Trend1",Trend1.UpdateCount,
Trend1.ScooterPosLeft,Trend1.ScooterPosRight,1,
"PenAverageValue");
```

HTScrollLeft() 历史

将趋势的起始时间值设置为比当前起始时间早趋势宽度的百分比。其结果是将图表日期/时间向左滚动给定的百分比。

语法

```
HTScrollLeft(Hist_Tag,Percent);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
Percent	表示图表要滚动的百分比的一个实数（0.0 到 100.0）。

实例

下面的语句将标记名 *Trend1* 的时间/日期滚动 10%：

```
HTScrollLeft("Trend1",10.0);
```

如果当前显示从中午 12：00：00 点开始、显示宽度为 60 秒，则新的趋势将在上午 11:59:54 开始（在函数执行后）。

HTScrollRight() 历史

将趋势的起始时间值设置为比当前起始时间晚于基于趋势宽度的百分比。其结果是将图表的日期/时间向右滚动给定的百分比。

语法

```
HTScrollRight(Hist_Tag,Percent);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
Percent	表示图表要滚动的百分比的一个实数（0.0 到 100.0）。

实例

下面的语句将标记名 *Trend1* 的时间/日期滚动 20%：

```
HTScrollRight("Trend1",20.0);
```

如果当前显示从中午 12：00：00 开始、显示宽度为 60 秒，则新的趋势将在下午 12：00：12 开始（在函数执行后）。

HTSelectTag() 历史

显示“**选择标记**”对话框，使操作员可以为指定的笔选择不同的标记名。此对话框仅列出标记名字典中的历史记录（选定“记录数据”选项）的已定义标记名。

语法 `HTSelectTag();`

备注 **HTSelectTag()** 显示所有已选定“**记录数据**”选项的标记。但是，可以使用浏览器的过滤器显示一组较小的标记名。例如所有以 "A" 开头的标记名。但是不能使用 **HTSelectTag()** 函数来显示标记名字典中的所有标记名，否则只会显示历史标记名。

实例 可在触动按扭动作 QuickScript 中使用下面的 QuickScript。QuickScript 将使标记浏览器显示在 WindowViewer 中。这样用户可以从列表中选择一个标记名。此标记名将被名为 HistTrend 的历史对象用作 1 号笔。

```
HTSetPenName( "HistTrend", 1, HTSelectTag( )  );
```

参见 **HTSetPenName()**

HTSetPenName() 历史

对趋势笔指定不同的标记名。

语法 **HTSetPenName**(Hist_Tag,PenNum,Tagname);

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
PenNum	表示笔号（从 1 到 8）的整型标记名或整数值，Tagname 是一个表示要赋予笔的新标记名的字符串。
Tagname	表示要分配给笔的新标记名的字符串。

备注 此 QuickScript 函数是在运行期间从分布式历史记录供应器添加标记名的唯一途径。

实例 在下面的语句中，*Trend1* 的 *Pen3* 将使用 *OutletPressure* 作为新的标记名。

```
HTSetPenName("Trend1",3,"OutletPressure");
```

在下面的语句中，*Trend1* 的 *TrendPen4* 将使用分布式标记名 *HistPrv1.Tag1* 作为标记名。

```
HTSetPenName("Trend1",TrendPen4,"HistPrv1.Tag1");
```

参见 **HTSelectTag();**

HTUpdateToCurrentTime() 历史

通过使结束时间等于当前时间来检索和显示数据。开始时间将等于结束时间减去图表的宽度。

语法

HTUpdateToCurrentTime(*Hist_Tag*);

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。

实例

下面的语句检索并显示当前时间下历史标记名“*Trend1*”的数据：

```
HTUpdateToCurrentTime("Trend1");
```

如果现在是下午 3:04，且趋势宽度为 60 秒，则新的结束时间将为下午 3:04。新的开始时间将为下午 3:03。

HTZoomIn() 历史

计算新图表的宽度和开始时间。如果趋势的 **.ScooterPosLeft** 为 0.0 而 **.ScooterPosRight** 为 1.0，那么新图表的宽度将等于旧图表宽度除以 2。新的开始时间将根据 *LockString* 的值进行计算。

语法

HTZoomIn(Hist_Tag,LockString);

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
LockString	代表缩放类型的字符串：

类型	描述
"StartTime"	使开始时间与缩放前相等
"Center"	使中心时间与缩放前相等
"EndTime"	使结束时间与缩放前相等

备注

如果指示器的位置不在两端，新图表的宽度将为 **.ScooterPosLeft** 和 **.ScooterPosRight** 之间的时间差。在这种情况下，将不使用 *LockString* 的值。最小的图表宽度为 1 秒。在缩放后，指示器的位置将置为 **.ScooterPosLeft**=0.0 和 **.ScooterPosRight**=1.0。

实例

下面的语句将显示缩小为原来的二分之一，并保持趋势标记名“*Trend1*”的起始时间不变。Trend1.ScooterPosLeft 等于 0.0， Trend1.ScooterPosRight 等于 1.0。如果缩放前的开始时间为下午 1:25:00，图表宽度为 30 秒（缩小后），开始时间将仍为 1:25:00，而图表宽度将变为 15 秒。

```
HTZoomIn("Trend1","StartTime");
```

HTZoomOut() 历史

计算新图表的宽度和开始时间。新图表的宽度等于旧图表宽度乘以 2。新的开始时间将根据 *LockString* 的值进行计算。

语法

```
HTZoomOut(Hist_Tag,LockString);
```

参数	描述
Hist_Tag	表示趋势名的历史趋势标记名。
LockString	代表缩放类型的字符串：

类型	描述
"StartTime"	使开始时间与缩放前相等
"Center"	使中心时间与缩放前相等
"EndTime"	使结束时间与缩放前相等

备注

指示器的位置对 **HTZoomOut** 没有影响，但在缩放后将置为 **.ScooterPosLeft**=0.0 和 **.ScooterPosRight**=1.0。

实例

下面的语句将显示放大为原来的二倍，并保持趋势标记名“Volume”的中心时间不变。如果在缩放前的开始时间为午 12:15:00、图表宽度为 30 秒，则缩放后的开始时间将变为 2:14:45，图表宽度将变为 60 秒，趋势的中心时间仍为 2:15:15。

```
HTZoomOut("Volume","Center");
```

InfoAppActive() 系统

测试应用程序是否处于活动状态。

语法

DiscreteResult=InfoAppActive(*AppTitle*);

参数	描述
AppTitle	表示要报告的应用程序的字符串。

实例

InfoAppActive("计算器") 将返回 0
{如果它不在运行}

注意：特定应用程序的标题可以通过 **InfoAppTitle()** 函数确定。

下列的 QuickScript 实例使用 **InfoAppActive** 检查系统中运行的任务列表。如果“记事本”任务正在运行，则返回结果 1，这样可避免第二次启动“记事本”。如果 **InfoAppActive** 返回值 0（记事本没有运行），那么记记本可以被启动。

```
IF InfoAppActive( InfoAppTitle( "Notepad" ) ) == 1 THEN
  ActivateApp  InfoAppTitle( "Notepad" );
ELSE
  StartApp  "Notepad";
ENDIF;
```

InfoAppTitle() 系统

返回应用程序的标题或者当前正在运行的指定程序的 Windows 任务列表名。

语法

MessageResult=**InfoAppTitle**(*“ProgramEXEName”*);

参数	描述
ProgramEXEName	表示要报告的程序的字符串。

实例

下面的语句将处理 ProgramEXEName“calc”并返回“Calculator”。实际的程序名为 calc.exe，不要在程序 EXE 名称字符中使用 ".exe"。

InfoAppTitle("calc") 将返回 "Calculator"

InfoAppTitle("excel") 将返回 "Microsoft Excel"

InfoDisk() 系统

返回指定的本地（或网络）磁盘驱动器信息。

语法

```
IntegerResult=InfoDisk("Drive",InfoType,Trigger);
```

参数	描述
Drive	表示驱动器字母的字符串或消息型标记名。
InfoType	表示信息类型的整数：

类型	描述
1	返回磁盘驱动器的总空间（以字节计）。
2	返回磁盘驱动器上的可用空间（以字节计）。
3	返回磁盘驱动器的总空间（以千字节计）。
4	返回磁盘驱动器上的可用空间（以千字节计）。

参数	描述
Trigger	每当 *Trigger* 值改变时执行 **InfoDisk** 函数。Trigger 可以是任意标记名（不仅仅限于系统变量）。此参数只用于动画链接中的表达式域，当 **InfoDisk()** 用于 QuickScript 中时，可以使用任意值作为占位符，因为此参数在 QuickScript 中无效。

备注

由 *Drive* 指定的磁盘驱动器的信息将返回给 *IntegerTag*。

实例

下面的语句每分钟执行一次并且返回当前值。如果用在值显示模拟动画链接中，它将每分钟刷新一次。

InfoDisk("C", 1, $Minute) 返回 233869345
{以字节表示的总空间}

InfoDisk("C", 2, $Minute) 返回 3238935
{以字节表示的可用空间}

InfoDisk("C", 3, $Minute) 返回 228388
{以千字节表示的总空间}

InfoDisk("C", 4, $Minute) 返回 3163
{以千字节表示的可用空间}

注释

1 千字节=1024 字节

注意：与其它使用单个字符的函数一样，如果提供给 **InfoDisk()** 函数的消息型标记名（用于 *Drive*）包含一个以上的字符，将只使用标记名的第一个字符。由于函数依赖于从操作系统中返回的消息，因此此值在包含大于 2G 驱动器的操作系统中可能会出现错误。

InfoFile() 系统

返回指定文件或子目录的信息。

语法

```
IntegerResult=InfoFile("Filename","InfoType",Trigger);
```

参数	描述
Filename	表示作用于实际字符串或消息型标记名的文件名的字符串。
InfoType	表示要检索的信息类型的整数：
Trigger	任意标记名。每当 *Trigger* 值改变时，执行 **InfoFile** 函数。此参数仅用于动画链接中的表达式域。当在 QuickScript 中使用 **InfoFile()** 时，可以使用任意值作为占位符，因为此参数在 QuickScript 中无效。

InfoType 类型：

类型	描述
1	*文件是否存在?* 如果文件名是一个实际文件，则返回 1。若文件名是一个子目录，则返回 2。若函数找不到文件，则返回 0。
2	文件大小（以字节计）。
3	文件日期/时间（自 1970 年 1 月 1 日起的相对秒数）。
4	与文件名描述相匹配的文件数。仅当使用通配符查找并且找到多个匹配文件时，才会返回大于 1 的值。

备注

由 *Filename* 指定的文件信息将返回给 *IntegerTag*。*Filename* 必须包括文件的完整路径，但可包括通配符 (*，?)。

实例

下面的语句每分钟执行一次并返回下列值：

InfoFile("c:\IT56\view.exe", 1, $Minute) 返回
1 {找到文件}

InfoFile("c:\InTouch\view.exe", 2, $Minute) 返回
634960 {文件大小}

InfoFile("c:\InTouch\view.exe", 3, $Minute) 返回
736701852 {自 1-1-70 以来的秒数}

InfoFile("c:\InTouch*.exe", 4, $Minute) 返回
17 {找到 17 个 EXE 文件}

InfoInTouchAppDir() 系统

返回当前 InTouch 应用程序目录。

语法

```
MessageResult=InfoInTouchAppDir();
```

备注

当前 InTouch 应用程序目录将返回给 *MessageResult*。

实例

```
InfoInTouchAppDir();
will return
"c:\ProgramFiles\FactorySuite\InTouch\DemoApp1\1024"
```

Int() 数学

返回小于或等于指定数字的下一个整数。

语法

```
IntegerResult=Int(Number);
```

参数	描述
Number	任意数字、实型或整型标记名。

备注

当处理负实数时，此函数将返回离零最远的整数。

实例

```
Int(4.7) 返回 4

Int(-4.7) 返回 -5
```

IOGetApplication() 其它

返回为特定访问名定义的应用程序名给指定的标记名。

语法

```
IOGetApplication("AccessName");
```

参数	描述
AccessName	要返回应用程序名的现有访问名。

备注

访问名可指定为文字字符串，或由其它 InTouch 标记名或函数提供的字符串值。下面的例子将访问名 **MyAccess1** 的应用程序名返回给标记名 **MyTag1**。

实例

```
MyTag1 = IOGetApplication("MyAccess1");
```

IOGetNode() 其它

返回为特定访问名定义的节点信息（地址）给指定的标记名。

语法 **IOGetNode**(*"AccessName"*);

参数	描述
AccessName	要返回节点信息的现有访问名。

备注 访问名可指定为文字字符串，或由其它 InTouch 标记名或函数提供的字符串值。下面的例子将访问名 **MyAccess1** 的节点信息返回给标记名 **MyTag1**。

实例

```
MyTag1 = IOGetNode("MyAccess1");
```

IOGetTopic() 其它

返回为特定访问名定义的主题名给指定的标记名。

语法 **IOGetTopic**(*"AccessName"*);

参数	描述
AccessName	要返回主题名的现有访问名。

备注 访问名可指定为文字字符串，或由其它 InTouch 标记名或函数提供的字符串值。下面的例子将访问名 **MyAccess1** 的主题名返回给标记名 **MyTag1**。

实例

```
MyTag1 = IOGetTopic("MyAccess1");
```

IOReinitialize() 其它

关闭所有的现有 I/O 对话并重新启动设置 I/O 对话的全过程。当执行此函数时，所有 I/O 点均会受影响。

语法

```
IOReinitialize();
```

备注 此函数将执行与 WindowViewer“**特别**”菜单上的“**重新初始化 I/O**”命令相同的操作。

IOSetAccessName() 其它

在运行期间修改访问名的*应用程序*或*主题*名部分，以允许实施 InTouch 的“热备份”策略。

语法

IOSetAccessName(“AccessName”, “NodeName”, “AppName”, “TopicName”);

参数	描述
AccessName	应用程序名和主题名值所指定到的现有访问名。实际字符串或消息型标记名。
NodeName	要指定的新节点名。实际字符串或消息型标记名。 要使用现有节点名，请输入不含字符串的 ""。要使用本地节点，请输入 " "。注意空格。
AppName	要指定的新应用程序名。实际字符串或消息型标记名。
TopicName	要指定的新主题名。实际字符串或消息型标记名。

备注

访问名、应用程序名和主题名可以指定为文字字符串，也可以是由其它 InTouch 标记或函数提供的字符串值。例如，访问名 **MyAccess1** 可以通过下面的语句改为指向“EXCEL”应用程序和“Sheet1”主题：

实例

```
IOSetAccessName("MyAccess1","MyComputer1","EXCEL",
"[Book1.xls]Sheet1");
```

或

```
Number = 1;
AccNameString = "MyAccess" + Text(Number, "#");
IOSetAccessName(AccNameString, "", "EXCEL",
"[Book1.xls]Sheet1");
```

如果为主题指定了一个空白字符串，那么访问名的当前应用程序值将被保留，其主题值将被更新。例如，下面的语句将问名 **MyAccess2** 的应用程序名改为“excel”，而不影响其当前主题值：

```
IOSetAccessName("MyAccess2", "", "excel", "");
```

同样，如果只为应用程序名指定了空白字符串，那么标记的当前主题值将被保留，其应用程序值将被更新。例如，下面的语句将标记名 **MyAccess3** 的主题改为“Sheet2”，而不影响其当前的应用程序名称值：

```
IOSetAccessName("MyAccess3", "","", "[Book1.xls]Sheet2");
```

注意：当处理 **IOSetAccessName()** 时，现有对话结束与新对话开始之间会有一段时间延迟。在此期间，任何试图插入或写入新主题的操作均将丢失。

IOSetItem() 其它

改变 I/O 型标记名 **.Reference** 域中的访问名和（或）项目。

语法

```
IOSetItem(“Tagname”,”AccessName”,”Item”);
```

参数	描述
Tagname	用引号括起的任意 InTouch IO 标记名。
AccessName	改变标记名的访问名。
Item	改变标记名的项目。

实例

```
IOSetItem(TagName, AccessName, Item)
```

标记名、访问名和项目值可指定为文字字符串，也可以由其它 InTouch 标记或函数提供的字符串值。例如，标记名 **MyTag1** 的 **.Reference** 域可以通过下面的语句改为指向“excel”访问名和“R1C1”项目：

```
IOSetItem("MyTag1", "[Book1.xls]excel", "R1C1");
```

或者：

```
Number = 1;
TagNameString = "MyTag" + Text(Number, "#");
IOSetItem(TagNameString, "[Book1.xls]excel", "R1C1");
```

如果 AccessName 和 Item 二者均指定了空白字符串 ("")，那么标记名将无效。例如，可使用下面的语句使标记名 **MyTag2** 无效：

```
IOSetItem("MyTag2", "", "");
```

如果只为项目指定了空白字符串，那么标记的当前项目值将被保留，访问名值被更新。例如，下面的语句将标记名 **MyTag3** 的访问名改为 "excel2"，但不影响其当前项目值：

```
IOSetItem("MyTag3", "[Book1.xls]excel2", "");
```

同样，如果只为访问名指定了一个空字符串，则标记名的当前项目值会被保留，而其访问名会被更新。例如，下面的语句将标记名 **MyTag3** 的项目名变为“R1C2”，但不影响其当前访问名值：

```
IOSetItem("MyTag4", "", "R1C2");
```

IOStartUninitConversations() 其它

当 WindowViewer 启动时，它会自动处理*初始化*请求来启动所有 I/O 对话。如果 I/O 服务程序未对 WindowViewer 的*初始化*请求作出回应，您可以通过在 QuickScript 中执行此函数来强制 WindowViewer 再次尝试建立 I/O 对话。

语法 `IOStartUninitConversations();`

备注 此函数执行与 WindowViewer“**特别**”菜单上的“**启动未初始化的对话**”命令相同的操作。

IsAnyAsynchFunctionBusy() 系统

用于找出是否有任何异步 QuickFunction 正在运行。使用此函数，您可以使调用异步 QuickFunction 的 QuickScript 等待所有其它异步 QuickFunction 完成处理。这允许 QuickScript 自我重新同步。

语法 `DiscreteTag=`**`IsAnyAsynchFunctionBusy`**`(`*`timeout`*`);`

参数	描述
DiscreteTag	是一个离散型标记名，其返回值如下所示： 如果函数在等候所有执行中的 QuickFunction 完成时超时，将返回值 1（真）给 ***DiscreteTag***。 如果没有异步 QuickFunction 运行，则立即返回值 0（假），否则 QuickFunction 将等待超时值到达。如果超时后没有异步 QuickFunctions 运行，则也会返回值 0。
Timeout	是一个表示等待秒数的整型值，用于确定是否有任何异步 QuickFunctions 正在运行。 值零 (0) 表示不等待。

实例 假如您要使用异步 QuickFunctions 连接到几个 SQL 数据库，您知道这需要 2 分钟来建立连接。首先，您需要执行一个异步 QuickFunction 来连接到 SQL 数据库。下一步，您将启动函数 **IsAnyAsynchFunctionBusy(120)，**使 SQL 有足够的时间在 QuickFunction 完成前建立连接。

但是，如果 2 分钟后连接仍未创建而且异步 QuickFunctions 仍忙于建立连接，则函数 **IsAnyAsynchFunctionBusy()** 会返回值 1（真）。您现在就可以显示一则错误信息，告诉操作员 SQL 连接失败。

使用下列窗口“**显示时**”QuickScript：

```
IF IsAnyAsynchFunctionBusy(120) == 1 THEN
   SHOW "SQL Connection Error Dialog";
ENDIF;
```

Log() 数学

返回自然对数。

语法

RealResult=**Log**(*Number*);

参数	描述
Number	任意数字、实型或整型标记名。

备注

计算 *Number* 的自然对数并返回给 **RealResult**。0 的自然对数未定义。

实例

Log(100) 返回 4.605...

Log(1) 返回 0

LogMessage() 其它

将用户自定义消息写入 Wonderware Logger。

语法

`LogMessage(“Message_Tag”);`

参数	描述
Message_Tag	记入 Wonderware Logger 的字符串。实际字符串或消息型标记名。

备注

这是用于 InTouch 脚本除错的一个功能强大的函数。通过在脚本中灵活使用 **LogMessage()** 函数，您可以判定执行 QuickScript 的次序脚本性能，并确定改变前后受 QuickScript 影响的标记名的值。记入 Wonderware Logger 的每条消息都会加上准确的日期和时间标签。

实例

```
LogMessage("Report Script is Running");
```

上面的语句将下列内容写入 Wonderware Logger：

```
94/01/14 15:21:14 WWSCRIPT Message:Report Script is
Running.
LogMessage(“The Value of MyTag is “ + Text(MyTag, “#”));
MyTag+MyTag + 10;
LogMessage(“The Value of MyTag is “ + Text(MyTag, “#”));
```

有关 Wonderware Logger 的详细信息，请参阅您的联机*《FactorySuite 系统管理员指南》*。

LogN() 数学

返回以 *n* 为底的 *x* 的*对数*值。

语法

`Result=LogN(Number,Base);`

参数	描述
Number	任意数字、实型或整型标记名。
Base	设置为底数的整数。数字或整型标记名。

备注

未定义以 1 为底的对数。

实例

LogN(8, 3) 返回 1.89279

如果 NumberTag 包含 3，BaseTag 包含 7，LogN(NumberTag, BaseTag) 将返回 0.564.。

Pi() 数学

返回 *Pi* 的值。

语法 `RealResult=`**`Pi`**`();`

实例 **Pi() 返回 3.1415926**

PlaySound() 其它

通过 Windows 声音设备（如已安装），播放 .wav 文件或由 Windows 注册表文件的 **[sounds]** 部分中指定的声波格式文件。

语法 **`PlaySound`**`("SoundName",Flags);`

参数	描述
SoundName	表示要播放的声音文件的字符串或消息型标记名。
Flags	Flags 可以是下列类型之一：

类型	描述
0	同步播放声音（缺省）
1	异步播放声音
2	不使用缺省声音。Playsound 是 .wav 文件的文件名。PlaySound 也可以接收 Windows 注册表 [Sounds] 部分的项目名。例如，如果 Windows 注册表文件中有一个如下所示的项目： **`MouseClick=C:\Sounds\Click.wav`** 您可以输入 MouseClick 作为 SoundName。
3	不用!
4	不用!
5-7	不用!
8	重复此声音直到下次调用 **PlaySound()** 函数为止。适用于 Windows 2000。
9	调用 **PlaySound()**，适用于 Windows NT。
16	不停止任何当前播放的声音。

实例 **`PlaySound ("`*`c:\horns.wav`*`",1);`**

注意：声音必须适合可用物理内存，并且需在已安装 wave 形式的音频设备驱动程序上播放。搜索声音文件的目录顺序如下：当前目录、Windows 目录、Windows 系统目录、PATH 中列出的目录。如果不能发现指定的 .wav 文件，请进入"**开始**"菜单，指向"**设置**"，单击"**控制面板**"，然后双击"**声音**"图标，设置您的缺省声音，如不能找到缺省声音，将不会播放声音。

PrintHT() 历史

可用于按钮上打印与指定历史趋势类型标记名相关联的历史趋势图。使用此函数时，历史趋势必须在屏幕上可见。

语法

```
PrintHT("Trend_Tag");
```

参数	描述
Trend_Tag	历史趋势标记名，实际字符串或消息型标记名。

备注

无

实例

```
PrintHT("HistTrend1");
```

PrintScreen() 其它

打印指定的屏幕。

语法

```
PrintScreen(ScreenOption, PrintOption );
```

参数	描述
ScreenOption	1 - View 的客户区域（无菜单） 2 - Window 区域（WindowViewer 的窗口中的内容，包括菜单）。 **注意：**无效选项（包括 0）缺省为客户区域。
PrintOption	1 – 最适合（保留图象长宽比不变，垂直或水平缩放） 2 – 纵向缩放（使图象根据页面纵向尺寸缩放 – 保留长宽比不变） 3 – 横向缩放（使图象根据页面横向尺寸缩放 – 保留长宽比不变） 4 – 全屏缩放（使图象充满整个页面 – 可以改变长宽比） **注意：**无效选项（包括 0）缺省为“最适合”。

备注

在任何模式下，View 区域外的弹出式窗口均会被裁剪。此外，其它“置前”应用程序，如置前模式下的调试程序，会在与 WindowViewer 窗口重叠时打印出来。

此 QuickScript 函数的工作原理类似于 PrintWindow() 函数，不同的是还会打印窗口数。您可以通过在 INTOUCH.INI 文件中添加下面的语句来控制 View 等待的时间：

```
PrintScreenWait=10000
```

此处，10000 表示等待的毫秒数。

字体用字体打印，对象是用位图方式绘制并打印的。具有白色背景且仅包含文字的窗口可以很快打印出来。打印彩色背景且包含许多对象的窗口将持续很长的时间。

注意：要确保窗口中的文本能正确打印，建议将所有要打印的窗口中的文本域设为“True Type”字体。

当打印按钮时，按钮上的文本中可能被“剪切”，这是因为按钮文本上使用的“系统”字体不是“True Type”字体。此外，“系统”字体用在打印机上与用在屏幕上略有不同。如果出现此情况，可以尝试扩大按钮。

如果打印机正用于打印报警，要使用 **PrintWindow()** 函数就需要第二台打印机。

如果窗口与分布式报警显示、组合框、文本框、单选钮、复选框等重叠，**PrinScreen()** 将不能正常打印。

PrintWindow() 其它

打印指定的窗口。

语法

`PrintWindow("Window",Left,Top,Width,Height,Options);`

参数	描述
Window	要打印的窗口名称，实际字符串或消息型标记名。
Left	以英寸为单位的浮点数，用于设置左边距；数字或实型标记名。
Top	以英寸为单位的浮点数，用于设置顶边距；数字或实型标记名。
Width	以英寸为单位的浮点数，用于设置打印输出的宽度。此参数可以取 0 以使用缺省的最大纵横比，或者可以是一指定的宽度；数字或实型标记名。
Height	以英寸为单位的浮点数，用于设置打印输出的高度。此参数可以取 0 以使用缺省的最大的纵横比，或者可以是一指定的高度；数字或实型标记名。
Options	仅当宽度和高度为 0 时，才使用离散值 0 或 1。如果 Options 为 1，窗口将在最大纵横比下以窗口尺寸的整数倍数打印；离散数字或离散型标记名。 如果 Options 为 0，窗口将以适合此页的最大纵横比打印。

注意：如果窗口包含位图，将 *Options* 置为 1，以免位图拉长。

备注

可以使用此函数对许多报告排队（在打印整个屏幕而不仅仅打印趋势图表时，建议用 **PrintWindow()** 函数代替 **PrintHT()** 函数，并且每次用户都应从历史趋势运行时对话框打印）。

当此函数运行时，WindowViewer 会将窗口载入“后台”内存区中。WindowViewer 将等待 10 秒钟，直至所有的 DDE 变量均被更新。随后此窗口被送至打印机。指定的窗口不一定要打开或可见才能打印。您可以通过在 INTOUCH.INI 文件中添加下面的语句来控制 View 等待的时间：

```
PrintWindowWait=10000
```

此处，10000 表示等待的毫秒数。

字体用字体打印，对象是用位图方式绘制并打印的。具有白色背景且仅包含文字的窗口可以很快打印出来。打印彩色背景且包含许多对象的窗口将持续很长的时间。

要确保窗口中的文本能正确打印，建议将所有要打印的窗口中的文本域设为“True Type”字体。

当打印按钮时，按钮上的文本中可能被“剪切”，这是因为按钮文本上使用的“系统”字体不是“True Type”字体。此外，“系统”字体用在打印机上与用在屏幕上略有不同。如果出现此情况，可以尝试扩大按钮。

如果打印机正用于打印报警，则需要用到第二台打印机来使用 **PrintWindow()** 函数。

如果窗口与分布式报警显示、组合框、文本框、单选钮、复选框等重叠，**PrintWindow()** 将不能正常打印。

如果窗口不处于活动状态，则窗口打印输出现在将显示列表框、组合框和其它窗口/AcitveX 控件（如活动状态下的标记浏览器）。如果窗口为活动窗口，则可以进行打印输出。

换句话说，打印按钮必须位于要打印的相同窗口内。如果打印按钮不在要打印的窗口上，打印输出时将丢失列表框、组合框和其它 ActiveX 控件。

为避免出现此问题，在使用 PrintWindow() 函数之前，请使用 **Show()**、**ShowAt()** 或 **almMoveWindow()** 函数将窗口激活。

实例

下面的“为真时”条件 QuickScript 在每天上午 8:30 打印三页报表：

条件

```
$Hour == 8 AND $Minute == 30
```

脚本主体

```
PrintWindow("1st Shift Summary",1,1,0,0,0);
PrintWindow("2nd Shift Summary",1,1,0,0,0);
PrintWindow("3rd Shift Summary",1,1,0,0,0);
```

如果窗口名称存在并且此窗口可进入打印队列，**PrintWindow()** 函数返回 1。否则返回 0。因此可以监视函数的状态。

```
Status=PrintWindow("Shift Summary",1,1,0,0,0);
```

Status 是一个可写入 1 或 0 的离散型标记名。

RecipeDelete()　配方

从指定配方模板文件中删除配方名。

语法

```
RecipeDelete("Filename","RecipeName");
```

参数	描述
FileName	函数所作用的配方模板文件；实际字符串或消息型标记名。
RecipeName	将被函数删除的指定配方模板文件中的配方。**RecipeLoad()**、**RecipeSave()** 和 **RecipeDelete()** 函数要求用户提供 RecipeName。**RecipeSelectRecipe()** 函数返回值给此参数。实际字符串或消息型标记名。

实例

下面的语句从 recfile.csv 文件中删除配方“Recipel”：

```
RecipeDelete("c:\recipe\recfile.csv", "Recipe1");
```

RecipeGetMessage()　配方

将错误代码写入模拟型标记名；将对应的错误代码消息写入消息型标记名。

语法

```
RecipeGetMessage(Analog_Tag,Message_Tag,Number);
```

参数	描述
Analog_Tag	不带引号或常数的实际整型或实型标记名。
Message_Tag	不带引号或字符串文字的实际整型或实型标记名。
Number	此参数设置返回给 **Message_Tag** 的最大字符串长度。InTouch 消息型标记名的最长可为 131 个字符。除非您已经减少了 InTouch 标记名字典中 **Message_Tag** 的最大字符串长度，否则此参数使用 131。此参数可以是一个常数或包含数字的整型标记名。

实例

通过在 InTouch 数据改变 QuickScript 中使用 **RecipeGetMessage()** 函数，相应的错误代码可以写入模拟型标记名，而关联的错误代码消息可以写入消息型标记名中：

```
Data Change Script Tagname[.field]:ErrorCode
Script body:RecipeGetMessage(ErrorCode, ErrorMessage,131);
```

每次模拟标记名 ErrorCode 的值改变时，将自动执行此 QuickScript。当执行此 QuickScript 时，**RecipeGetMessage()** 函数会读取标记名 ErrorCode 的当前数值，并将与该数值关联的消息返回给标记名 ErrorMessage。

```
ErrorCode = RecipeLoad
("c:\App\recipe.csv","Unit1","cookies");
RecipeGetMessage(ErrorCode, ErrorMessageTag, 131);
```

RecipeLoad() 配方

将指定的配方加载到指定的标记名单元中。

语法

`RecipeLoad("Filename","UnitName","RecipeName");`

参数	描述
FileName	函数将作用到的配方模板文件名；*FileName* 可以是字符串常数或包含配方模板文件的**消息**型标记名。
UnitName	函数调用的指定配方模板文件中的特定单元名称。**RecipeLoad()** 函数需要用户提供 *UnitName*。**RecipeSelectUuit()** 函数返回值给此参数。*UnitName* 可以是包含单元名称的字符串常数或**消息**标记名。
RecipeName	函数使用的指定配方模板文件中的特定配方名称。**RecipeLoad()**、**RecipeSave()** 和 **RecipeDelete()** 函数要求用户提供 RecipeName。**RecipeSelectRecipe()** 函数返回值给此参数。*RecipeName* 可以是包含配方名称的字符串常数或消息标记名。

实例

下面的语句将为配方 **Recipe1**（在 recfile.csv 文件中）定义的值加载到为 Unitl 定义的标记名中：

```
RecipeLoad("c:\recipe\recfile.csv", "Unit1", "Recipe1");
```

RecipeSave() 配方

将最新创建的配方或对现有配方的更改保存到指定的配方模板文件中。

语法

```
RecipeSave("Filename","UnitName","RecipeName");
```

参数	描述
FileName	函数将作用到的配方模板文件名；*FileName* 可以是字符串常数或包含配方模板文件的消息型标记名。
UnitName	函数调用的指定配方模板文件中的特定单元名称。**RecipeLoad()** 函数需要用户提供 *UnitName*。**RecipeSelectUuit()** 函数返回值给此参数。UnitName 可以是包含单元名称的字符串常数或消息标记名。
RecipeName	函数使用的指定配方模板文件中的特定配方名称。**RecipeLoad()**、**RecipeSave()** 和 **RecipeDelete()** 函数要求用户提供 RecipeName。**RecipeSelectRecipe()** 函数返回值给此参数。*RecipeName* 可以是包含配方名称的字符串常数或消息标记名。

实例

下面的例子将保存对 recfile.csv 文件中的配方“**Recipe3**”所做的更改。如果 **Recipe3** 不存在于 recfile.csv 文件中，则会创建一个。此数值用于设置为 Unit2 定义的标记名值。

```
RecipeSave("c:\recipe\recfile.csv", "Unit2", "Recipe3");
```

RecipeSelectNextRecipe() 配方

选择当前在配方模板文件中定义的下一个配方名称。

语法

```
RecipeSelectNextRecipe("Filename", RecipeName, Number);
```

参数	描述
FileName	函数将作用到的配方模板文件名；实际消息型标记名。
RecipeName	函数使用的指定配方模板文件中的特定配方名称。**RecipeLoad()**、**RecipeSave()** 和 **RecipeDelete()** 函数要求用户提供 RecipeName。**RecipeSelectRecipe()** 函数返回值给此参数。没有引号的实际消息型标记名或文字字符串。
Number	如果函数必须用填入字符参数，该域将设置返回给参数的最大字符串长度。在 InTouch 中，字符串（消息型）标记名最长可为 131 个字符。此参数一般设为 131，除非您减小了 InTouch 标记名的最大字符串长度。数字或整型标记名。

实例

下面的语句使系统读取 *RecipeName* 标记名的当前值并返回文件中的下一个配方。如果 *RecipeName* 的值为空或找不到，将返回此文件的第一个配方。如果 *RecipeName* 当前包含文件的最后一个配方名，它将返回并保持不变（配方按创建顺序保存）。

```
RecipeSelectNextRecipe("c:\recipe\recfile.csv",
RecipeName, 131);
```

RecipeSelectPreviousRecipe() 配方

选择当前在配方模板文件中定义的前一个配方名称。

语法

```
RecipeSelectPreviousRecipe("Filename",RecipeName,Number);
```

参数	描述
FileName	函数将作用到的配方模板文件名；实际消息型标记名。
RecipeName	函数使用的指定配方模板文件中的特定配方名称。**RecipeLoad()**、**RecipeSave()** 和 **RecipeDelete()** 函数要求用户提供 RecipeName。**RecipeSelectRecipe()** 函数返回值给此参数。没有引号的实际消息型标记名或文字字符串。
Number	如果函数必须用填入字符参数，该域将设置返回给参数的最大字符串长度。在 InTouch 中，字符串消息型标记名最长可达 131 个字符。此参数一般设为 131，除非您减小了 InTouch 标记名的最大字符串长度。数字或整型标记名。

实例

下面的语句使系统读取 *RecipeName* 标记名的当前值并返回文件中的前一个配方名称。此返回的字符串将保存到 *RecipeName* 中并覆盖当前值。如果 *RecipeName* 的值为空或找不到，则会返回文件的最后一个配方。如果 *RecipeName* 当前包含文件的第一个配方名称，它将返回并保持不变（配方按创建顺序保存）。

```
RecipeSelectPreviousRecipe("c:\recipe\recfile.csv",
RecipeName, 131);
```

RecipeSelectRecipe() 配方

选择当前在配方模板文件中定义的特定配方名称。

语法 **RecipeSelectRecipe**("Filename",RecipeName,Number);

参数	描述
FileName	函数将作用到的配方模板文件名；实际消息型标记名。
RecipeName	函数使用的指定配方模板文件中的特定配方名称。**RecipeLoad()**、**RecipeSave()** 和 **RecipeDelete()** 函数要求用户提供 RecipeName。**RecipeSelectRecipe()** 函数返回值给此参数。没有引号的实际消息型标记名或文字字符串。
Number	如果函数必须用填入字符参数，该域将设置返回给参数的最大字符串长度。在 InTouch 中，字符串消息型标记名最长可达 131 个字符。此参数一般设为 131，除非您减小了 InTouch 标记名的最大字符串长度。数字或整型标记名。

实例 下面的语句打开“**选择配方**”对话框：

```
RecipeSelectRecipe("c:\recipe\recfile.csv", RecipeName,
131);
```

一旦从对话框中选定配方，其名称将返回给标记名 *RecipeName*。

RecipeSelectUnit() 配方

选择当前配方值所加载到的标记名单元。

语法 **RecipeSelectUnit**("Filename", UnitName,Number);

参数	描述
FileName	函数将作用到的配方模板文件名；实际消息型标记名。
UnitName	函数调用的指定配方模板文件中的特定单元名称。**RecipeLoad()** 函数需要用户提供 *UnitName*。**RecipeSelectUuit()** 函数返回值给此参数。没有引号的实际消息型标记名或文字字符串。
Number	如果函数需要填入字符参数，此参数将设置返回给参数的最大字符串长度。在 InTouch 中，字符串（消息型）标记名最长可为 131 个字符。此参数一般设为 131，除非您减小了 InTouch 标记名的最大字符串长度。数字或整型标记名。

实例 下面的语句打开“**选择单元**”对话框：

```
RecipeSelectUnit("c:\recipe\recfile.csv", UnitName, 131);
```

一旦选定单元，其名称将返回给标记名 *UnitName*。

注意：RecipeSelectRecipe() 和 **RecipeSelectUnit()** 函数均需要结合 **RecipeLoad()** 函数使用。

有关合并函数的详细信息，请参阅《*配方管理器用户指南*》中的“合并配方函数”一节。

ReloadWindowViewer() 系统

允许用户控制重新载入 WindowViewer 的函数过程。

语法 `ReloadWindowViewer();`

备注 此函数将自动更新 WindowViewer。当未使用“自动更新网络应用程序开发 (NAD)”功能时，此函数可用于更新应用程序。此函数可结合 **$ApplicationChanged** 使用，以确定 NAD 更新的时间，然后在不中断 View 节点操作的情况下更新该节点。在使用“**通知客户**”命令时，操作员可能需要使更新延迟到稍后的时间。此函数可用在触动按钮动作 QuickScript 中。因此，操作员可以在方便时动态更新 WindowViewer。此函数可有效代替 **RestartWindowViewer()** 函数。

参见 **$ApplicationChanged**

RestartWindowViewer() 系统

允许用户控制关闭和重新启动 WindowViewer 的函数过程。

语法 `RestartWindowViewer();`

备注 此函数将关闭并自动重新启动 WindowViewer。当未使用“自动更新网络应用程序开发 (NAD)”功能时，此函数可用于更新应用程序。此函数可结合 **$ApplicationChanged** 使用，以确定 NAD 更新的时间，然后通过关闭并重新启动 WindowViewer 来更新 View 节点。在使用“**通知客户**”命令时，操作员可能需要使更新延迟到稍后的时间。此函数可用在触动按钮动作 QuickScript 中。因此，操作员可以在方便的时候自动关闭并重新启动 WindowViewer。此函数通常被 **ReloadWindowViewer()** 取代，后者可以在不关闭 Viewer 的情况下更新 View 节点。

参见 **$ApplicationChanged; ReloadWindowViewer()**

Round() 数学

将实数舍入为指定的精度。

语法 `RealResult=Round(Number,Precision);`

参数	描述
Number	任意数字、实型或整型标记名。
Precision	设置用于舍入数字的精度；数字、实型或整型标记名。

备注 *Precision* 参数设置了舍入 *Number* 的精度。

实例

Round(4.3, 1) 返回 4

Round(4.3, .01) 返回 4.30

Round(4.5, 1) 返回 5

Round(-4.5, 1) 返回 -4

Round(106, 5) 返回 105

Round(43.7, .5) 返回 43.5

参见 **Trunc()**

SendKeys 其它

将键发送到应用程序。对接收应用程序来说，这些键就好象是从键盘上输入的。此项功能可用于给应用程序输入数据或给应用程序发送命令。**SendKeys** 语句可使用大部分的键盘键。每个键用一个或多个字符表示，例如，“A”表示字母“A”，{ENTER} 表示“回车”键。

语法

```
SendKeys KeySequence;
```

参数	描述
KeySequence	任意键序列或消息型标记名。

备注

要指定一个以上的键，需要将每个字符的代码串联起来。例如，要指定美元符号 ($) 后跟 (b)，请输入 $b。下表列出的每个键盘键的有效发送键。

键	代码	键	代码
BACKSPACE	{BACKSPACE}or {BS}	HOME	{HOME}
BREAK	{BREAK}	INSERT	{INSERT}
CAPSLOCK	{CAPSLOCK}	LEFT	{LEFT}
DELETE	{DELETE} or {DEL}	NUMLOCK	{NUMLOCK}
DOWN	{DOWN}	PAGE DOWN	{PGDN}
END	{END}	PAGE UP	{PGUP}
ENTER	{ENTER} or ~ (tilde)	PRTSC	{PRTSC}
ESCAPE	{ESCAPE} or {ESC}	RIGHT	{RIGHT}
F1	{F1}*	TAB	{TAB}
		UP	{UP}

* 所有的功能键以相同的方式输入。

特殊键（SHIFT、CTRL 和 ALT）有自己的键码：

键	代码
SHIFT	+ (plus)
CTRL	^ (caret)
ALT	% (percent)

实例

如果同时使用两个特的键，则需要使用第二组括号。下面的语句用于在按住 ALT 键和 CTRL 键的同时再按下 P 键。

```
SendKeys "^(%(p))";
```

ActivateApp 命令可优先于别的命令执行，以将击键值传递给适当的应用程序。

下面的语句将使计算机运行 Excel 并且发送组合键 CTRL+P(此组合键将运行一个预先定义的打印宏命令)。

```
ActivateApp "Microsoft Excel";
SendKeys "^(p)";
```

或者，在 WindowViewer 中使用下面的语句打开“**登录**”对话框：

```
SendKeys "%(SYL)";
```

在标有 HELP 的按钮中，您可以使用下面的 QuickScript：

```
SendKeys "{F1}";
```

注意：Microsoft 对 Windows 硬件提取层的增强可能可以使此函数在某些个人计算机上保持运行。

SetPropertyD() GOT

指定要在运行时写入的属性的离散值。

语法

```
[ErrorNumber=]SetPropertyD("ControlName.Property",
DiscreteTag);
```

参数	描述
ControlName	窗口控件名，如 ChkBox_1，或报警对象名，如 AlmObj_1。
.Property	窗口控件或报警对象属性。 有关这些属性的详细信息，请参阅第 2 章“点域”。
DiscreteTag	当函数执行时，用于保存所写入值的离散值或离散型标记名。典型用法： 0 = 禁用控件 1=启用控件

备注 有关所返回错误号的列表，请参阅附录 A。

参见 **GetPropertyD(), GetPropertyI(), GetPropertyM(), SetPropertyI(), SetPropertyM()**

SetPropertyI() GOT

指定要在运行时写入的属性的整型值。

语法

```
[ErrorNumber=]SetPropertyI("ControlName.Property", Integer);
```

参数	描述
ControlName	窗口控件名，例如 *ChkBox_1*；或报警对象名，例如 AlmObj_1。
.Property	窗口控件或报警对象属性。 有关这些属性的详细信息，请参阅第 2 章“点域”。
整型	数字或整型标记名。

备注 有关所返回错误号的列表，请参阅附录 A。

参见 **GetPropertyD(), GetPropertyI(), GetPropertyM(), SetPropertyD(), SetPropertyM()**

SetPropertyM() GOT

指定要在运行时写入的属性的消息型值。

语法

```
[ErrorNumber=]SetPropertyM("ControlName.Property",
"MessageTag");
```

参数	描述
ControlName	窗口控件名，例如 *ChkBox_1*；或报警对象名，例如 AlmObj_1。
.Property	窗口控件或报警对象属性。 有关这些属性的详细信息，请参阅第 2 章“点域”。
MessageTag	将写入 *ControlName* 属性的字符串。实际字符串或消息型标记名。

备注 有关所返回错误号的列表，请参阅附录 A。

参见 **GetPropertyD(), GetPropertyI(), GetPropertyM(), SetPropertyI(), SetPropertyI()**

Sgn() 数学

确定值的符号（正、零或负）。

语法

```
IntegerResult=Sgn(Number);
```

参数	描述
Number	任意数字、实型或整型标记名。

备注 如果输入值为正数，则结果为 1；输入值为负数，返回 –1；输入值为 0，返回 0。

实例

Sgn(425) 返回 1

Sgn(0) 返回 0

Sgn(-37.3) 返回 -1

Show 其它

显示指定的窗口（窗口名必须用引号括起）。

语法

Show "*Window*";

参数	描述
Window	要显示的窗口名。实际字符串或消息型标记名。

备注

Window 必须与现有窗口或要创建的窗口名匹配。在运行期间，如果窗口不存在，WindowViewer 会忽略此语句。如果窗口名改变，则必须在 QuickScript 中相应改变。

注意：如果使用消息型标记名作为隐藏窗口的参数，将会出现下面的错误消息：

“需要窗口名 - 必须是字符串表达式”

为避免再次出现此消息，请在消息型标记名前面加上加号 (+)。例如：

```
DIM Test AS message;
Test – "MyWindow1";
Show ""+Test;
```

实例

```
Show "Alarm Summary Window";
```

注意：如果 QuickScript 只隐藏或显示窗口，建议使用 **Show** *Window* 或 **Hide** *Window* 链接。如果使用了这些函数但窗口名改变，InTouch 且自动做出改变。

参见

Hide, Hideself, ShowAt(), ShowHome, ShowTopLeftAt()

ShowAt() 其它

指定窗口显示时的水平和垂直象素位置。

语法

`ShowAt("Window",Horiz,Vert);`

参数	描述
Window	窗口名称；实际字符串或消息型标记名。
Horiz	水平像素位置；可以是任意数字或整型标记名。
Vert	垂直像素位置；可以是任意数字或整型标记名。

备注

当窗口打开时，它将位于水平和垂直座标的中心。如果其边界超出屏幕，则窗口不能居中放置，而是根据边界进行最佳调整。

实例

在下面的语句中，*100* 表示水平像素位置，*200* 表示垂直像素位置：

```
ShowAt("Window Name",100,200);
```

您也可以创建模拟量输入对象并将其链接到内存标记名，如 *TagHoriz* 和 *TagVert*，以在运行期间动态地改变窗口位置。在此情况下，应输入下面的语句：

```
ShowAt("Boiler Room 7 Details",TagHoriz,TagVert);
```

下面的语句确定对象的像素位置，内部标记名 **$ObjHor** 和 **$ObjVer** 可分配给用以显示当前选定对象的位置的模拟输出链接。通过使用链接到对象或按钮的 QuickScript 中的 **$ObjHor** 和 **$ObjVer** 标记名，可在此对象或按钮上居中显示窗口。

```
ShowAt("Window Name",$ObjHor,$ObjVer);
```

参见

$ObjHor, $ObjVer, ShowTopLeftAt(), Show, ShowHome, Hide, Hideself

ShowHome 其它

显示“起始”窗口。起始窗口是选定在 WindowViewer 启动时自动打开的窗口（起始窗口在“**WindowViewer 属性- 起始窗口**”属性页中选定）。

语法 `ShowHome;`

参见 **Show, ShowAt(), ShowTopLeftAt(), ShowHome, Hide, Hideself**

ShowTopLeftAt() 其它

指定窗口显示时其左上角的水平和垂直象素位置。

语法 `ShowTopLeftAt("Window",Horiz,Vert);`

参数	描述
Window	窗口名称；实际字符串或消息型标记名。
Horiz	水平像素位置；可以是任意数字或整型标记名。
Vert	垂直像素位置；可以是任意数字或整型标记名。

备注 当窗口打开时，其左上角将位于水平和垂直座标的相交处（屏幕最左上角的像素位置是 0，0）。此函数的工作原理与 **ShowAt()** 函数完全相同，不同的是它还控制窗口出现的左上角位置。

注意：此函数会将窗口载入内存，然后将其移至新位置。无论窗口如何移动，原始位置（在 WindowMaker 中保存窗口时）保留不变。

参见 **Show, ShowAt(), ShowHome(), Hide, Hideself**

Sin() 数学

返回给定角（以度表示）的*正弦*值。

语法 `Result=Sin(AngleNumber);`

参数	描述
AngleNumber	角的度数；任意数字、实型或整型标记名。

备注 计算 *Number* 的正弦并返回给 *Result*。

实例

```
Sin(90) 返回 1
Sin(0) 返回 0
wave = 100 * Sin (6 * $second);
```

参见 **Cos(), Tan(), ArcCos(), ArcSin(), ArcTan()**

SPCConnect() SPC

此函数结合自动数据采集数据集使用。在自动数据采集数据集开始采集数据之前，必须使用此函数向 SPC 指出用户所在节点。

语法 `SPCConnect(“User”,”Password”);`

参数	描述
User	数据库用户名；实际字符串或消息型标记名。
Password	用户口令；实际字符串或消息型标记名。

备注 执行此函数将使用户与数据库连接并且启动使用该用户 ID 的自动数据采集数据集。如果数据库没有定义口令，您可以使用下列 QuickScript：

实例

```
SPCConnect(“User1”, “”);
```

参见 **SPCDisconnect()**

SPCDatasetDlg() SPC

此函数用于在 WindowViewer 中显示“ **SPCPro 数据集配置**”对话框。可添加或删除新数据集。此函数没有任何参数或返回值。

当 WindowViewer 运行时，当前数据集和产品信息将灰白显示，某些字段将不能更改。

在 WindowViewer 运行时可以添加新的数据集和产品。

如果添加并保存了新的数据集和产品：

在自动采集模式下运行时，自动采集周期会重新启动，在数据集初始化阶段可能会有数据丢失。

建议您在运行时使用 **NewProduct** SPC DDE 项目来添加产品，而不要通过 **SPCDatasetDlg()** 函数，该函数会阻止重新启动自动采集模式。

语法

```
SPCDatasetDlg();
```

备注

执行此函数将在 WindowViewer 中打开“**SPCPro 数据库配置**”对话框。

实例

```
SPCDatasetDlg();
```

参见

SPCSelectDataset()

SPCDisconnect() SPC

此函数用于断开代理与 SPC Pro 数据库的连接。当使用此函数时，所有指定给已断开代理的数据集将停止采集。

语法

```
SPCDisconnect();
```

备注

执行此函数可断开用与数据库的连接并停止所有自动数据采集数据集。

实例

```
SPCDisconnect();
```

参见

SPCConnect()

SPCDisplayData() SPC

此函数的设计可方便地滚动图表到任何日期和时间。您可以使用标记名来监视 SPC 数据搜索的状态。在指定的时间内，如果 SPC 找到数据，Status 将包含 **0**，否则将包含 **1**。

语法

```
[Status=]SPCDisplayData("Dataset", "DateString",
"TimeString", RangeInHours );
```

参数	描述
Dataset	表示实际数据集名称；实际字符串或消息型标记名。
DateString	以 mm/dd/yy 格式表示的日期；实际字符串或消息型标记名。
TimeString	以 hh:mm:ss 格式表示的时间；实际字符串或消息型标记名。
RangeInHours	表示要显示的数据的小时数；可以是任意数字或整型标记名。

实例

```
StatusTag = SPCDisplayData("Dataset", "DateString",
"TimeString" , RangeInHours);
```

SPCLocateScooter() SPC

此函数的设计可方便地滚动指示器到任何有效样本号。数据集中定义的指示器标记名将用 X-Bar 样本值更新。将 **SampleNumber** 设为 **0** 将隐藏/禁用指示器。

语法

```
SPCLocateScooter( "Dataset", SampleNumber );
```

参数	描述
Dataset	表示实际数据集名称；实际字符串或消息型标记名。
SampleNumber	表示任意有效样本号可以是任意数字或整型标记名。

参见 **SPCMoveScooter()**

SPCMoveScooter() SPC

此函数的设计可方便地滚动指示器到任何有效样本号。数据集中定义的指示器标记名将用 X-Bar 样本值更新。

语法

```
SPCMoveScooter( "Dataset", IncrementValue );
```

参数	描述
Dataset	表示实际数据集名称；实际字符串或消息型标记名。
IncrementValue	表示任意有效数字。正值向前滚动，负值向后滚动。可以是任意数字或整型标记名。

参见 **SPCLocateScooter()**

SPCSaveSample() SPC

保存手工输入样本。此函数与 **SPCSetMeasurement** 函数结合使用。

语法

```
SPCSaveSample("Dataset");
```

参数	描述
Dataset	实际数据集名称；实际字符串或消息型标记名。

备注 执行此函数将强制样本输入到指定的数据集名称。对于样本中的测量值，将使用手工输入变量的当前值。通过 DDE 标记名 **MI_Mx** (x =从 1 到 25 的测量数字)或使用 **SPCSetMeasurement()** 函数（测量 1 到 300），可设置输入值。您需要重新启动 DDE 对话来更新 **MI_Mx** 标记的值。

参见 **SPCSetMeasurement()**

SPCSelectDataset() SPC

打开一个用户可以从中选择间接数据集的对话框。

语法 `DatasetName=`**`SPCSelectDataset`**`()`

备注 此 QuickScript 可打开“**选择数据集**”对话框。

一旦选定数据集名，此函数会将其返回给 *DatasetName* 标记名。此选择也可用于改变间接数据集中的 *DatasetName*。

参见 **SPCSelectProduct(), SPCDatasetDlg()**

SPCSelectProduct() SPC

打开一个对话框，用户可以从中选择给定数据集中的产品。

语法 `ProductName=`**`SPCSelectProduct`**`(`*`Dataset`*`);`

备注 执行此 QuickScript 可打开“**选择产品**”对话框。

一旦选定产品名，此函数会将其返回给 *ProductName* 标记名。此选择也可用于改变 *Dataset* 中的已采集产品。

参见 **SPCSelectDataset(), SPCSetProductDisplayed(), SPCSetProductCollected()**

SPCSetControlLimits() SPC

允许方便地以手工或事件驱动方式输入控制图的控制极限值。

语法 `SPCSetControlLimits("Dataset",XUCL,XLCL);`

参数	描述
Dataset	包含实际数据集名称；实际字符串或消息型标记名。
XUCL	表示图表中用于 UCL 的值；数字或实型标记名。
XLCL	表示图表中用于 LCL 的值；数字或实型标记名。

备注 这些测量通过执行 **SPCSaveSample()** 保存到样本中。

参见 **SPCSaveSample(), SPCSetRangeLimits(), SPCSetSpecLimits(), SPCSetMeasurement()**

SPCSetMeasurement() SPC

通过执行 QuickScript，允许方便地以手工或事件驱动方式输入模拟测量值。

语法 `SPCSetMeasurement("Dataset",Measurement,Value);`

参数	描述
Dataset	包含实际数据集名称；实际字符串或消息型标记名。
Measurement	表示测量号（从 1 至 300）；可以是任意数字或整型标记名。
Value	写入指定测量号的值；任意数字或整型标记名。

备注 如果将所有测量均设为保存数据到数据库，请使用 **SPCSaveSample()**。您需要重新启动 DDE 对话来更新 **MI_Mx** 标记的值。

参见 **SPCSaveSample()**

SPCSetProductCollected() SPC

更改在指定的数据集中采集的产品。

语法 `SPCSetProductCollected("Dataset","Product");`

参数	描述
Dataset	包含实际数据集名称；实际字符串或消息型标记名。
Product	包含需要采集数据的产品名称；实际字符串或消息型标记名。

备注 此函数不更改所显示的产品。通过使用此函数来采集数据，使用 **SPCSetProductDisplayed()** 函数来显示数据，可以实现为一个产品采集数据而为另一产品显示数据。

实例

```
SPCSetProductCollected("Data5838", "Widgets");
```

参见 **SPCSelectProduct(), SPCSetProductDisplayed()**

SPCSetProductDisplayed() SPC

更改在指定的数据集中显示的产品。

语法 `SPCSetProductDisplayed("Dataset","Product");`

参数	描述
Dataset	包含实际数据集名称；实际字符串或消息型标记名。
Product	包含所要显示数据的产品名称；实际字符串或消息型标记名。

实例

```
SPCSetProductDisplayed("ADataSetName", "AProductName");
```

参见 **SPCSelectProductt(), SPCSetProductCollected()**

SPCSetRangeLimits() SPC

允许方便地以手工或事件驱动方式输入范围图的控制限。

语法 `SPCSetRangeLimits("Dataset",RUCL,RLCL);`

参数	描述
Dataset	包含实际数据集名称；实际字符串或消息型标记名。
RUCL	表示用于范围图表的 UCL 的值；数字或实型标记名。
RLCL	表示用于范围图表的 LCL 的值；数字或实型标记名。

参见 **SPCSetControlLimits(), SPCSetSpecLimits()**

SPCSetSpecLimits() SPC

允许方便地以手工或事件驱动方式输入控制图的规格限。

语法 `SPCSetSpecLimits("Dataset",XUSL,XLSL);`

参数	描述
Dataset	包含实际数据集名称；实际字符串或消息型标记名。
XUSL	表示用于图表的 USL 的值；数字或实型标记名。
XLSL	表示用于图表的 LSL 的值；数字或实型标记名。

参见 **SPCSetControlLimits(), SPCSetRangeLimits()**

SQLAppendStatement() SQL

使用 *String* 的内容来追加 SQL 语句。函数将返回错误。

语法

```
[ResultCode=]SQLAppendStatement(ConnectionID,
"SQLStatement");
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
SQLStatement	要追加的实际语句。

实例

```
ResultCode=SQLAppendStatement(ConnectionID,
"where tablename.columnname=(any value or string)");
```

参见 **SQLConnect(), SQLClearStatement()**

SQLClearParam() SQL

清除指定参数的值。在调用 **SQLExecute()** 函数前，必须重新调用 SQLSetParam 。

语法

```
[ResultCode=]SQLClearParam(SQLHandle,ParameterNumber);
```

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	要修改的 SQL 语句中的实际参数。

参见 **SQLPrepareStatement(), SQLExecute()**

SQLClearStatement() SQL

释放与 *SQLHandle* 指定的语句相关联的资源。

语法

`[ResultCode=]`**`SQLClearStatement`**`(`*`ConnectionID,SQLHandle`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。

参见

SQLConnect(), SQLPrepareStatement()

SQLClearTable() SQL

删除数据库表格中的所有记录，但保留表格。

语法

`[ResultCode=]`**`SQLClearTable`**`(`*`ConnectionID,"TableName"`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。
TableName	您要访问的数据库表格名称。

实例

下面的语句将删除 BATCH1 表格中的所有记录：

```
ResultCode=SQLClearTable(ConnectionID,"BATCH1");
```

参见

SQLConnect(), SQLClearStatement()

SQLCommit() SQL

SQLCommit() 命令定义了一组操作命令的结束。在 **SQLTransact()** 命令和 **SQLCommit()** 命令之间执行的一组命令称为事务集。事务集的处理方式和单个事务的一样。在发出 SQLTransact() 命令后，所有后续操作必须等待 SQLCommit() 命令发出后才能提交给数据库。

语法

```
[ResultCode=]SQLCommit(ConnectionID);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。

注意：在编写包含 **SQLCommit()** 命令的 QuickScript 时务必小心。因为处理时间与事务集命令数目呈倍数关系，所以含有多条命令的脚本在执行时会很慢。

实例

```
ResultCode = SQLTransact( ConnectionID);
ResultCode = SQLInsertPrepare( ConnectionID, TableName,
BindList, SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertEnd( ConnectionID, SQLHandle );
ResultCode = SQLCommit( ConnectionID);
{数据库保留 3 个插入}
```

参见

SQLRollback(), SQLTransact(), SQLCommit()

SQLConnect() SQL

将 InTouch 连接到 *ConnectString* 中指定的数据库。

语法

```
[ResultCode=]SQLConnect(ConnectionID,"ConnectString");
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。
ConnectString	用以识别数据库和用于 **SQLConnect()** 函数中的任何附加登录信息的字符串。

实例

下面的语句将连接到 IBM OS/2 数据库管理器和名为 SAMPLE 的数据库：

```
[ResultCode=]SQLConnect(ConnectionID,"DSN=OS2DM;DB=SAMPLE")
;
```

此函数返回一个值给变量 *ConnectionID*，该变量将用作所有后续 SQL 函数中的参数。

ConnectString 可识别数据库系统和任何附加登录信息。它以下列格式输入：

```
"DSN=data source name[;attribute=value
[;attribute=value]...]"
```

不同的数据库需要不同的属性。QELIB 可识别所有数据库的以下属性：

属性	值
DSN	在 Microsoft ODBC 管理器中配置的数据源名称。
DLG	当启用 (DLG=1) 时，会显示一个对话框，允许您输入连接字符串信息。
DRV	若数据源名 (DSN) 不在连接字符串中出现，可以使用此值，以和 InTouch 4.1 版本的 SQL 访问兼容。QELIB 会将其改为数据源名称。
UID	登录 ID。
PWD	口令。
MODIFYSQL	QELIB 用此来确保应用程序中的 SQL 和数据库中的 SQL 相兼容。当置为 1（缺省值）时，数据库驱动程序需要 ODBC 兼容**语法**，此语法可根据基本数据库系统的需要进行修改。当置为 0 时，数据库驱动程序需要并支持基本数据库系统本身的**语法**。这使得您可以继续使用那些用 QELIB 1.0 数据库驱动程序支持的 SQL 开发的应用程序。

REREADAFTERUPDATE	当启用（置为 1） 时，QELIB 在记录更新后会从数据库重新读取记录。这对得到自动更新列（如，时间标签）的正确值是很有用的。
REREADAFTERINSERT	当启用（置为 1）时，QELIB 在插入记录后重新读取记录。这对得到自动更新列（如，时间标签）的正确值是很有用的。

有关特定数据库所支持属性的详细信息，请参阅您的联机《*InTouch SQL Access 用户指南*》。

参见 **SQLDisconnect()**

SQLCreateTable() SQL

利用指定的表格模板中的参数在数据库中建立表格。表格模板（在文件 SQL.DEF 中定义）决定数据库表格的结构。

语法

[ResultCode=]**SQLCreateTable**(*ConnectionID,TableName,* TemplateName);

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
TableName	您要访问的数据库表格名称。
TemplateName	您要使用的模板定义的名称。

实例

下面的语句通过 *TEMPLATE* 中定义的列名和类型创建一个名为 *BATCH1* 的表格：

```
ResultCode=SQLCreateTable(ConnectionID,"BATCH1",
"TEMPLATE");
```

当在 QuickScript 中输入以引号括起的参数时，例如 "Parameter1"，将使用整个字符串；如果没有用引号括起，那么 Parameter1 将假定为标记名，系统就会从 InTouch 标记名字典中访问标记名 Parameter1 的值。例如：

"c:\main\file" vs. location

此处，location 是 InTouch 消息型标记名。

"c:\main\file" 是一个文字字符串

参见 **SQLConnect()**

SQLDelete() SQL

删除一条或多条记录。

语法

```
[ResultCode=]SQLDelete(ConnectionID,TableName,WhereExpr);
```

注意：SQLDelete() 函数不能包含空的 *WhereExpression*。

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
TableName	要访问的数据库表格名。
WhereExpression	为表格的任意一行定义真假条件。此函数仅从表格中提取条件为真的行。表达式必须为下列格式： ColumnName *comparison_operator* expression. **注意：**如果列为字符数据类型，则表达式必须用单引号括起。

实例

下面的例子将选择共 name 列包含值 EmployeeID 的所有行：

```
name='EmployeeID'
```

下面的例子将选择包含从 100 到 199 的 partno 的所有行：

```
partno>=100 and partno<200
```

下面的例子将选择其 temperature 列包含大于 350 的值的所有行：

```
temperature>350
```

下面的语句从 *BATCH1* 表中删除批号等于 65 的所有记录：

```
ResultCode=SQLDelete(ConnectionID,"BATCH1","lotno=65");
```

参见

SQLConnect()

SQLDisconnect() SQL

断开用户与数据库的连接。

语法 `[ResultCode=]`**`SQLDisconnect`**`(`*`ConnectionID)`*`;`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。

参见 **SQLConnect()**

SQLDropTable() SQL

废除表格。

语法 `[ResultCode=]`**`SQLDropTable`**`(`*`ConnectionID,TableName`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
TableName	您要访问的数据库表格名称。

实例 下面的语句将废除 *BATCH1* 表，此命令一旦执行，表格名将不可识别并且不响应任何命令。

```
ResultCode=SQLDropTable(ConnectionID,"BATCH1");
```

参见 **SQLConnect()**

SQLEnd() SQL

在 SQLSelect() 函数后使用，以释放存储结果表格所用的资源。

语法

```
[ResultCode=]SQLEnd(ConnectionID);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。

参见 **SQLConnect(), SQLSelect()**

SQLErrorMsg() SQL

检索与特定 *ResultCode* 关联的文本错误消息。*ErrorMsg* 是与 *ResultCode* 关联的 InTouch 内存消息型标记名（最长 131 个字符）。

语法

```
SQLErrorMsg(ResultCode);
```

参数	描述
ResultCode	大多数 SQL 函数返回的整型变量。若函数执行成功，其值为 0；若函数执行不成功，其值为负。 有关这些代码的详细信息，请参阅附录 A“SQL 脚本函数疑难排除”。

实例

```
ErrorMsg=SQLErrorMsg(ResultCode);
```

参见 **SQLConnect()**

SQLExecute() SQL

执行 SQL 语句。如果语句是 Select 语句，*BindList* 参数将指定*绑定表*的名称，以将数据库中的列与标记名关联。如果*绑定表*为空，则没有形成标记名关系。例如，SQL 语句可以是“Create View”、“Insert”等。返回函数中将返回错误。如果语句已经“预备”，则应传递从预备语句返回的语句句柄。如果语句未“预备”，语句的句柄应为零 (0)。

语法

```
[ResultCode=]SQLExecute(ConnectionID,BindList,SQLHandle);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。
BindList	定义要使用的 InTouch 标记名以及与之关联的数据库列。
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。

参见 **SQLConnect(),SQLPrepareStatement()**

注意：没有预备的语句，**SQLExecute()** 只能调用一次。如果语句已经预备，则函数可多次调用。

SQLFirst() SQL

选定由最后一个 **SQLSelect()** 函数创建的结果表格中的首条记录。调用此命令前必须先执行 **SQLSelect()** 函数。

语法

```
[ResultCode=]SQLFirst(ConnectionID);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。

参见 **SQLConnect(), SQLSelect()**

SQLGetRecord() SQL

从当前选择缓冲区中检索由 *RecordNumber* 指定的记录。

语法

`[ResultCode=]`**`SQLGetRecord`**`(`*`ConnectionID,Record Number`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
RecordNumber	要检索的实际记录数值。

实例

```
ResultCode=SQLGetRecord(ConnectionID,3);
```

参见

SQLConnect()

SQLInsert() SQL

根据提供的绑定表中的标记名值，将一条新记录插入参考表中。*BindList* 参数定义使用哪些 InTouch 标记名及与之关联的数据库列。

语法

`[ResultCode=]`**`SQLInsert`**`(ConnectionID,TableName,BindList);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 `SQLConnect` 函数赋给每个数据库的编号 (ID)。
TableName	您要访问的数据库表格名称。
BindL*ist*	定义要使用的 InTouch 标记名以及与之关联的数据库列。

实例

下面的语句将 List1 中指定的标记名值，将一条新记录插入到 ORG 表格中：

```
ResultCode=SQLInsert(ConnectionID,"ORG","List1");
```

注意：可使用下面的三个函数来代替标准 **SQLInsert()** 函数，以将记录快速地插入到文件中。**SQLInsert()** 函数是将一次性插入并结束语句的操作。因此，若再次调用 **SQLInsert()** 函数，整个操作过程将重新进行一遍。这与下面的三个函数相比将要更长的时间。下面的三个函数将这些步骤分开，以便一旦您执行一个 **SQLInsertPrepare()** 函数，就可以随心所欲地执行多个 **SQLInsertExecute()** 函数，然后在结束时执行一个 **SQLInsertEnd()** 函数。

SQLInsertEnd() SQL

释放语句。

语法

`[ResultCode=]`**`SQLInsertEnd`**`(`*`ConnectionID,SQLHandle`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
SQLHandle	当使用 **SQLPrepareStatement()** 函数时， 由 SQL 返回的整数值。

参见 **SQLConnect(), SQLPrepareStatement()**

SQLInsertExecute() SQL

执行已预备的语句。

语法

`[ResultCode=]`**`SQLInsertExecute`**`(`*`ConnectionID,BindList,SQLHandle`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
BindList	定义要使用的 InTouch 标记名以及与之关联的数据库列。
SQLHandle	当使用 **SQLPrepareStatement()** 函数时， 由 SQL 返回的整数值。

参见 **SQLConnect(), SQLPrepareStatement()**

SQLInsertPrepare() SQL

创建并准备一个要执行的插入语句。插入语句不会处理。*SQLHandle* 参数是一个整型标记名，它包含语句执行后的返回值。

语法

```
[ResultCode=]SQLInsertPrepare
(ConnectionID,TableName,BindList,SQLHandle);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
TableName	要访问的数据库表格名。
BindList	定义要使用的 InTouch 标记名以及与之关联的数据库列。
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。

参见 **SQLConnect(), SQLPrepareStatement()**

SQLLast() SQL

选择由最近一个 **SQLSelect()** 函数创建的结果表格中的最后一条记录。调用此命令前必须先执行 **SQLSelect()** 函数。

语法

```
[ResultCode=]SQLLast(ConnectionID);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。

实例

```
ResultCode=SQLLast(ConnectionID);
```

参见 **SQLConnect(), SQLSelect()**

SQLLoadStatement() SQL

读取 *FileName* 中包含的语句。在这一点上，此语句类似于由 **SQLSetStatement()** 函数创建的语句，并且能通过 **SQLAppStatement()** 函数来追加记录，或由 **SQLExeucte()** 函数执行。每个文件只允许包含一条语句。但是，如果尚未调用 **SQLPrepareStatement()** 或 **SQLExeucte()** 函数，就可以用 **SQLAppendStatement()** 在语句中追加内容。

语法

[ResultCode=]**SQLLoadStatement**(*ConnectionID,FileName*);

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
FileName	包含消息的文件名。

备注

预备由 **SQLSetStatement** 或 **SQLLoadStatement** 函数创建的 SQL 语句。将返回语句的句柄。

实例

```
ResultCode=SQLLoadStatement(ConnectionID,
"C:\InTouchAppname\SQL.txt")
```

```
SQL.txt = Select ColumnName from TableName where
ColumnName>100;
```

参见

SQLConnect(), SQLAppendStatement(), SQLExecute(), SQLPrepareStatement

SQLManageDSN() SQL

运行 Microsoft ODBC 管理器配置程序。可用于添加、删除和修改所有的数据源名称。

语法

SQLManageDSN(*ConnectionID*);

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。

SQLNext() SQL

选择由最后的 SQLSelect() 函数建立的结果表的下一条记录。调用此命令前必须先执行 **SQLSelect()** 函数。

语法 [ResultCode=]**SQLNext**(*ConnectionID*);

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。

实例

```
ResultCode=SQLNext(ConnectionID);
```

参见 **SQLConnect(), SQLSelect()**

SQLNumRows() SQL

指示有多少行满足最后的 SQLSelect() 函数中指定的条件。例如，如果使用 *WhereExpression* 来选定所有 AGE 列值为 45 的行，则返回的行数可以是 40 或 4000。这可以确定下一步要执行的函数。

语法 **SQLNumRows**(*ConnectionID*);

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。

实例 下面的语句将选定行的行数返回给整型标记名 NumRows：

```
NumRows=SQLNumRows(ConnectionID);
```

参见 **SQLConnect()**

SQLPrepareStatement() SQL

SQLPrepareStatement() 预备由函数 **SQLSetParam()** 使用的现有 SQL 语句。可通过使用 **SQLSetStatement()** 或 **SQLLoadStatement()** 来建立语句。将返回语句的句柄。

语法

`[ResultCode=]`**`SQLPrepareStatement`**`(`*`ConnectionID,SQLHandle`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。

备注

必须调用匹配的 **SQLClearStatement()** 来释放系统资源。

要正确调用存储过程，请使用下面的语句：

```
ResultCode = SQLSetStatement(ConnectionID,
 "exec sp_MyStoredProc");
```

实例

```
ResultCode=SQLPrepareStatement(ConnectionID,SQLHandle);
```

参见

SQLConnect(), SQLSelect(), SQLSetStatement(), SQLLoadStatement()

SQLPrev() SQL

选择由最后的 **SQLSelect()** 函数创建的结果表的前一条记录。

语法

`[ResultCode=]`**`SQLPrev`**`(`*`ConnectionID`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。

备注

调用此命令前必须先执行 **SQLSelect()** 函数。

实例

```
ResultCode=SQLPrev(ConnectionID);
```

参见

SQLConnect(), SQLSelect()

SQLRollback() SQL

SQLRollback() 命令会反转或“回滚”最近提交的事务集。事务集是由 **SQLTransact()** 命令和 **SQLCommit()** 命令，或 **SQLRollback()** 命令之间发出的一组命令。事务集的处理方式和单个事务的一样。在发出 **SQLTransact()** 命令后，所有后续操作必须等待 **SQLCommit()** 命令发出后，才能提交给数据库。

语法

`[ResultCode=]`**`SQLRollback`**`(`*`ConnectionID,)`*`;`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。

实例

```
ResultCode =SQLTransact( ConnectionID);
ResultCode = SQLInsertPrepare( ConnectionID, TableName,
BindList, SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertEnd( ConnectionID, SQLHandle );
ResultCode =SQLRollback( ConnectionID);
{ 忽略 Transact() 后面的所有命令，数据库不变}
```

参见

SQLCommit(), SQLTransact()

SQLSelect() SQL

通知数据库从表格中检索信息。当执行 **SQLSelect()** 函数时，会在内存中创建一个临时结果表，其中含有可用 **SQLFirst()**、**SQLLast()**、**SQLNext()** 和 **SQLPrev()** 函数浏览的记录。

注意：每次在 **SQLSelect()** 之后使用 **SQLEnd()** 来释放存储结果表所用的资源。

语法

```
[ResultCode=]SQLSelect(ConnectionID,TableName,BindList,
WhereExpr,OrderByExpr);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
TableName	要访问的数据库表格名。
BindList	定义所用的 InTouch 标记名和数据库。
WhereExpression	为表格的任意一行定义真假条件。此命令只从表格中提取条件为真的行。表达式必须为下列格式：

ColumnName *comparison_operator* expression.

注意：如果列为字符数据类型，则表达式必须用单引号括起。 ColumnName comparison_operator expression.

下面的例子将选择共 name 列包含值 EmployeeID 的所有行：

name='EmployeeID'

下面的例子将选择包含从 100 到 199 的 partno 的所有行：

partno>=100 and partno<200

下面的例子将选择其 temperature 列包含大于 350 的值的所有行：

temperature>350

WhereExpression 实例：

WhereExpr - 内存消息型标记
OrderByExpr - 内存消息型标记
Speed_Input - 内存实型 - 用户输入模拟
Serial_Input - 内存消息型 - 用户输入字符串

模拟实例

```
WhereExpr = “Speed = “ + text
(Speed_Input,”#.##”);
```

因为 Speed_Input 是一个数字，它必须转化为文本，然后再与 where 表达式字符串连接。

字符串实例

```
WhereExpr = “Ser_No = ‘“ +
Serial_input + “’”;
```

因为 Serial_Input 是一个字符串，它必须用单引号括起，例如：WhereExpr = “Ser_No=’125gh’”;

使用 like 语句的字符串实例

```
WhereExpr = “Ser_No like ‘” + “125%’”
```

当使用 Like 时，可使用 % 字符作为通配符。

使用 And 的字符串和模拟实例

```
WhereExpr = “Ser_No = ‘“ + Serial_input +
“’” + “ and “ + “Speed = “ +
text(Speed_Input,”#.##”);
```

```
OrderByExpr = “”;
```

如果次序无关紧要，可使用上面显示的空字符串。

SQLSelect 使用 WhereExpr 标记名

```
ResultCode =
SQLSelect(Connect_Id,TableName,
BindList,
```

```
WhereExpr,OrderByExpr);
```

```
Error_msg = SQLErrorMsg( ResultCode );
```

函数中内建的 SQLSelect WhereExpr

```
ResultCode =
SQLSelect(Connect_Id,TableName,
BindList,
```

```
“Ser_No = ‘“ + Serial_input + “’”,
OrderByExpr);
```

```
Error_msg = SQLErrorMsg( ResultCode );
```

每次在完成 **SQLSelect()** 后调用 SQLEnd(*Connect_Id*) 函数。如果不调用 **SQLEnd()**，资源将不能释放，应用程序将用尽内存。

OrderByExpression 定义排序的列和方向。只可使用列名来排序，并且表达式的格式必须如下所示：

ColumnName [ASC|DESC]

下面的实例将对选定表格的“manager”列按递增顺序进行排序：

"manager ASC"

您也可以按照下面的表达式格式对多个列进行排序：

ColumnName [ASC|DESC],

ColumnName [ASC|DESC]

下一个例子将对按照“temperature”列递增和“time”列递减的顺序对选定的表格进行排序：

"temperature ASC, time DESC"

实例

下面的语句使用列类型包含“Cookie”的绑定表 List1，从 BATCH 表格中取出记录。它将显示按“amount”列的递增顺序和“sugar”列的递减顺序排序的信息：

```
ResultCode=SQLSelect(ConnectionID,"BATCH","List1",
"type='cookie'","amount ASC, sugar DESC");
```

下面的语句没有为 WhereExpression 和 OrderByExpression 指定值，因而可以选择数据库中的所有数据：

```
ResultCode=SQLSelect(ConnectionID,"BATCH","List1","","");
```

参见

SQLFirst(), SQLConnect(), SQLLast(), SQLNext(), SQLPrev(), SQLEnd(), SQLSelect()

SQLSetParamChar() SQL

将指定参数的值设置为指定的字符串。**SQLSetParamChar()** 函数可在执行前多次调用，从而使参数值设为全部发送值的串接。0（零）长度将被忽略。

语法

`[ResultCode=]`**`SQLSetParamChar`**`(SQLHandle,ParameterNumber,ParameterValue,MaxLen);`

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。
MaxLen	与此参数关联的最大列尺寸。此设置决定参数是可变字符类型还是长可变字符类型。如果 MaxLen 小于或等于数据库所允许的最大字符串，那么此参数为可变字符类型。否则为长可变字符类型。

实例

`ResultCode=SQLSetParamChar(SQLHandle,ParameterNumber,ParameterValue,MaxLen);`

参见

SQLPrepareStatement()

SQLSetParamDate() SQL

将指定日期参数的值设为指定的字符串。

语法

`[ResultCode=]`**`SQLSetParamDate`**`(SQLHandle,ParameterNumber,ParameterValue);`

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。

实例

`ResultCode=SQLSetParamDate(SQLHandle,ParameterNumber,ParameterValue);`

参见

SQLPrepareStatement()

SQLSetParamDateTime() SQL

将指定日期–时间参数的值设为指定的字符串。

语法

```
[ResultCode=]SQLSetParamDateTime(SQLHandle,ParameterNumber,
ParameterValue,Precision);
```

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。
Precision	要指定的日期-时间值长度。这是要使用的 *ParameterValue* 中的字符数。

实例

```
ResultCode=SQLSetParamDateTime(SQLHandle,ParameterNumber,Pa
rameterValue,Precision);
```

参见

SQLPrepareStatement()

SQLSetParamDecimal() SQL

将指定的小数参数值设为指定的字符串。*Precision* 是数值位数，*Scale* 是小数点右边的位数。

语法

```
[ResultCode=]SQLSetParamDecimal(SQLHandle,ParameterNumber,
ParameterValue,Precision,Scale);
```

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。
Precision	要指定的日期-时间值长度。这是要使用的 *ParameterValue* 中的字符数。
Scale	小数点右边的位数。

实例

```
ResultCode=SQLSetParamDecimal(SQLHandle,ParameterNumber,
ParameterValue,Precision, Scale);
```

参见

SQLPrepareStatement()

SQLSetParamFloat() SQL

将指定的参数值设为指定的 *ParameterValue*。

语法

```
[ResultCode=]SQLSetParamFloat(SQLHandle,ParameterNumber,
ParameterValue);
```

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。

实例

```
ResultCode=SQLSetParamFloat(SQLHandle,ParameterNumber,
ParameterValue);
```

参见

SQLPrepareStatement()

SQLSetParamInt() SQL

将指定的参数值设为指定的 *ParameterValue*。

语法

```
[ResultCode=]SQLSetParamInt(SQLHandle,ParameterNumber,
ParameterValue);
```

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。

实例

```
ResultCode=SQLSetParamInt(SQLHandle,ParameterNumber,
ParameterValue);
```

参见

SQLPrepareStatement()

SQLSetParamLong() SQL

将指定的参数值设为指定的 *ParameterValue*。

语法

```
[ResultCode=]SQLSetParamLong(SQLHandle,ParameterNumber,
ParameterValue);
```

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。

实例

```
ResultCode=SQLSetParamLong(SQLHandle,ParameterNumber,
ParameterValue);
```

参见 **SQLPrepareStatement()**

SQLSetParamNull() SQL

将指定的参数值设为 NULL。

语法

[ResultCode=]**SQLSetParamNull**(SQLHandle,ParameterNumber, ParameterType,Precision,Scale);

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterType	指定参数的数据类型：

类型	描述
字符	可填空格的固定长度字符串
变量字符	可变长度字符串
十进制	
小数	BCD 码
整型	4 字节有符号整数
短整型	2 字节有符号整数
浮点	4 字节浮点
双精度浮点	8 字节浮点
日期时间	4 字节日期时间值
日期	2 字节日期时间值
时间	2 字节日期时间值
无类型	无数据类型

参数	描述
Precision	十进制值的精度、字符的最大长度，或日期-时间值的字节长度。
Scale	十进制值的比例。仅当适用参数设为空时，才需要使用此值。

备注

如果此参数已完成 **SQLSetParam()** 的调用，则可使用“无类型”。

实例

```
ResultCode=SQLSetParamNull(SQLHandle,ParameterNumber,
ParameterType,Precision,Scale);
```

参见

SQLPrepareStatement()

SQLSetParamTime() SQL

将指定的时间参数值设为指定的字符串。

语法

`[ResultCode=]`**`SQLSetParamTime`**`(SQLHandle,ParameterNumber, ParameterValue);`

参数	描述
SQLHandle	当使用 **SQLPrepareStatement()** 函数时，由 SQL 返回的整数值。
ParameterNumber	语句中的实际参数数目。
ParameterValue	实际设置值。

实例

`ResultCode=SQLSetParamTime(SQLHandle, ParameterNumber,`*`ParameterValue);`*

参见

SQLPrepareStatement()

SQLSetStatement() SQL

在已建立的连接 *ConnectionID* 上，根据 *SQLStatement* 的内容创建一个 SQL 语句缓冲区。每个 *ConnectionID* 可以有一个 SQL 语句缓冲区。函数将返回错误。

语法

[ResultCode=]**SQLSetStatement**(*ConnectionID,SQLStatement*);

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
SQLStatement	实际语句，参看下面的例子。

实例

```
ResultCode=SQLSetStatement(ConnectionID,"Select LotNo,
LotName from LotInfo");
```

在下面的例子中，**SQLHandle** 设为零，因此不必在执行语句前调用 SQLPrepare(Connect_Id, SQLHandle)。由于 **SQLhandle** 不是由 **SQLPepare** 建立的，要正确结束此选择，使用 **SQLEnd** 函数，而不要使用 **SQLClearStatement()** 函数。

```
SQLSetStatement( Connect_Id, “Select  Speed, Ser_No from
tablename where Ser_No =’” +   Serial_input  + “’”);
SQLExceute(Connect_Id,0);
```

在下面的例子中，**SQLhandl**e 由 **SQLPrepareStatement** 创建并用在 **SQLExceute** 函数中。要结束此选择语句，请使用 **SQLClearStatment** 来释放资源和 **SqlHandle**。

```
SQLSetStatement( Connect_Id, “Select  Speed, Ser_No from
tablename where Ser_No =’” +   Serial_input  + “’”);
SQLPrepareStatement(Connect_Id,SQlHandle);
SQLExceute(Connect_Id,SqlHandle);
SQLSetStatement( Connect_Id, “Select  Speed, Ser_No from
tablename where Ser_No =’” +   Serial_input  + “’”);
SQLPrepareStatement(Connect_Id,SQlHandle);
SQLExceute(Connect_Id,SqlHandle);
```

参见

SQLConnect()

SQLTransact() SQL

SQLTransact() 命令定义一组操作命令的开始。在 **SQLTransact()** 命令和 **SQLCommit()** 命令之间执行的一组命令称为事务集。事务集的处理方式和单个事务的一样。在发出 **SQLTransact()** 命令后，所有后续操作必须等待 **SQLCommit()** 命令发出后才能提交给数据库。

语法

```
[ResultCode=]SQLTransact(ConnectionID);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 `SQLConnect` 函数赋给每个数据库的编号 (ID)。

注意：在编写包含 **SQLTransact()** 命令的 QuickScript 时务必小心。因为处理时间与事务集命令数目呈倍数关系，所以含有多条命令的脚本在执行时会很慢。

参见

SQLCommit(), SQLRollback()

SQLUpdate() SQL

修改记录以用当前标记名值来更新记录。

语法

```
[ResultCode=]SQLUpdate(ConnectionID,TableName,BindList,
WhereExpr);
```

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 **SQLConnect** 函数赋给每个数据库的编号 (ID)。
TableName	要访问的数据库表格名。
BindList	定义要使用的 InTouch 标记名以及与之关联的数据库列。
WhereExpression	为表格的任意一行定义真假条件。此函数仅从表格中提取条件为真的行。表达式必须为下列格式： ColumnName *comparison_operator* expression. **注意：**如果列为字符数据类型，则表达式必须用单引号括起。 下面的例子将选择共 name 列包含值 EmployeeID 的所有行： **name='EmployeeID'** 下面的例子将选择包含从 100 到 199 的 partno 的所有行： **partno>=100 and partno<200** 下面的例子将选择其 temperature 列包含大于 350 的值的所有行： **temperature>350**

实例

下面的语句将 BATCH 表格中批号为 65 的所有记录更新为绑定表“List1”中指定的标记名的当前值。

```
ResultCode=SQLUpdate(ConnectionID,"BATCH","List1","lotno=65
");
```

注意：请确保所有的记录均为唯一。如果表格中存在相同的记录，则两者都将被更新。

参见

SQLConnect()

SQLUpdateCurrent() SQL

取出当前所选记录并用任何新的 InTouch 值更新它。下面的实例将更新当前选定的记录。

语法

`[ResultCode=]`**`SQLUpdateCurrent`**`(`*`ConnectionID`*`);`

参数	描述
ConnectionID	由用户创建的内存整型标记名，用于存储由 SQLConnect 函数赋给每个数据库的编号 (ID)。

实例

`ResultCode=SQLUpdateCurrent(ConnectionID);`

参见

SQLConnect()

Sqrt() 数学

使 InTouch 自动计算语句后面的值的平方根。

语法

`RealResult=`**`Sqrt`**`(`*`Number`*`);`

参数	描述
Number	任意数字、实型或整型标记名。

实例

`AnalogTag1=Sqrt(AnalogTag2);`

StartApp 系统

自动启动另一 Windows 应用程序。

语法

`StartApp "AppName";`

参数	描述
AppName	要启动的实际应用程序名，例如 Wordpad.exe。

建议输入带 .EXE 扩展名的程序名。如果可执行程序支持的话，也可输入命令行参数。

长文件名无效，但是下面使用 Dos 格式的长文件名的例子有效。

如果长文件名是：C:\Program files\Microsoft Office\Office\Excel。使用 C:\Progra~1\Micros~2\Office\Excel（Dos 格式）。

```
StartApp "C:\Progra~1\Micros~2\Office\Excel";
```

如果使用 Winfile 而且所有文件细节均显示在窗口中的中间列，将显示 Dos 格式的文件路径。

实例

下面的语句将启动 Microsoft Windows Wordpad 程序。

```
StartApp "Wordpad.exe";
```

参见

ActivateApp()

StringASCII() 字符串

返回指定消息型标记名的首字符的 ASCII 值。

语法

`IntegerResult=`**`StringASCII`**`("`*`Char`*`");`

参数	描述
Char	字母数字字符或消息型标记名。

备注

将 *Char* 中的首字符的 ASCII 值返回给 *IntegerResult*。当执行此函数时，只测试或影响单个字符。如果提供给 *StringASCII* 的消息型标记名包含一个以上的字符，则只会测试第一个字符。

实例

StringASCII("A") 返回 65

StringASCII("A Mixer is Running") 返回 65

StringASCII("a mixer is running") 返回 97

参见

StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringChar() 字符串

返回指定 ASCII 码的对应字符。

语法

```
MessageResult=StringChar(ASCII);
```

参数	描述
ASCII	ASCII 码或整型标记名。

备注

将由 *ASCII* 指定的 ASCII 字符返回给 *MessageResult*。此函数的一个用处是给消息型标记添加键盘上通常没有的特殊字符。

实例

```
ControlString=MessageTag+StringChar(13)+StringChar(10);
```

在 *MessageTag* 的结尾加上 [回车(13)] 和 [换行(10)] 并将其传递给 *ControlString*。插入 32-126 范围可显示字符之外的字符，对于创建外设（例如，打印机或调制解调器）的控制代码非常有用。

此函数通常用于 SQL 命令。Where 表达式有时要求字符串值带有双引号，因此使用 **StringChar**(34)。

参见

StringASCII(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringFromIntg() 字符串

将整型值转换成其它形式的字符串。

语法

```
MessageResult=StringFromIntg(Number,Base);
```

参数	描述
Number	要转换的数字。可以是任意数字或整型标记名。
Base	用于转换的底。可以是任意数字或整型标记名。

备注

Integer 转换成指定的 *base*，结果存储在 *MessageResult* 中。

实例

StringFromIntg(26, 2) 返回 "11010"

StringFromIntg(26, 8) 返回 "32"

StringFromIntg(26, 16) 返回 "1A"

参见

StringASCII(), StringChar(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringFromReal() 字符串

将实型值转换成浮点数或指数形式的字符串。

语法 MessageResult=**StringFromReal**(*Number,Precision,"Type"*);

参数	描述
Number	根据指定的精度和类型进行转换，结果保存在 *MessageResult* 中。数字或实型标记名。
Precision	指定要显示小数位数。可以是任意数字或整型标记名。
Type	决定显示方法：

类型	描述
"f"	以浮点表示法显示。
"e"	以小写“e”的指数形式显示。
"E"	以大写“E”的指数形式显示。

实例

StringFromReal(263.355, 2,"f") 返回 "263.36"

StringFromReal(263.355, 2,"e") 返回 "2.63e2"

StringFromReal(263.55, 3,"E") 返回 "2.636E2"

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()**

StringFromTime() 字符串

将时间值（1970 年 1 月 1 日起，以秒为单位）转换成指定的字符串表示法。时间值应为 UTC 相当时间（自 Jan-01-1970 GMT 起的秒数）。返回的值反映本地时间。

语法

```
MessageResult=StringFromTime(SecsSince1-1-70,StringType);
```

参数	描述
SecsSince1-1-70	转换成指定的字符串类型并且结果存储在 *MessageResult* 中。
StringType	决定显示方法：

类型	描述
1	以 Windows 控制面板中设置的格式显示日期（类似于为 $DateString 的显示）
2	以 Windows 控制面板中设置的格式显示时间（类似于 $TimeString 的显示）
3	显示 24 个字符的字符串，表示日期和时间："Wed Jan 02 02:03:55 1993"
4	以短格式显示星期几："Wed"
5	以长格式显示星期几："Wednesday"

实例

StringFromTime(86400, 1) 返回**"1/2/70"**

StringFromTime(86400, 2) 返回 **"12:00:00 AM"**

StringFromTime(86400, 3) 返回 **"Fri Jan 02 00:00:00 1970"**

StringFromTime(86400, 4) 返回 **"Fri"**

StringFromTime(86400, 5) 返回 **"Friday"**

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringFromTimeLocal() 字符串

将时间值（1970 年 1 月 1 日起，以秒为单位）转换成指定的字符串表示法。返回的值反映本地时间。

语法

MessageResult=**StringFromTimeLocal**(*SecsSince1-1-70, StringType*);

参数	描述
SecsSince1-1-70	转换成指定的字符串类型并且结果存储在 *MessageResult* 中。
StringType	决定显示方法：

类型	描述
1	以 Windows 控制面板中设置的格式显示日期（类似于为 $DateString 的显示）
2	以 Windows 控制面板中设置的格式显示时间（类似于 $TimeString 的显示）
3	显示 24 个字符的字符串，表示日期和时间："Wed Jan 02 02:03:55 1993"
4	以短格式显示星期几："Wed"
5	以长格式显示星期几："Wednesday"

实例

StringFromTimeLocal(86400, 1) 返回"1/2/70"

StringFromTimeLocal (86400, 2) 返回 "12:00:00 AM"

StringFromTimeLocal (86400, 3) 返回 "Fri Jan 02 00:00:00 1970"

StringFromTimeLocal (86400, 4) 返回 "Fri"

StringFromTimeLocal (86400, 5) 返回 "Friday"

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringInString() 字符串

返回 *SearchFor* 在 *Text* 中首次出现的位置。

语法

```
IntegerResult=StringInString("Text","SearchFor",StartPos,
CaseSens);
```

参数	描述
Text	查找 *SearchFor* 的出处，如果找到多处 *SearchFor*，则将第一次出现的位置返回给 *IntegerTag*。实际字符串或消息型标记名。
SearchFor	要在文本中查找的字符串。实际字符串或消息型标记名。
StartPos	一个用来确定 *Text* 中开始查找位置的整数。可以是任意数字或整型标记名。
CaseSens	决定查找是否区分大小写（0=否，1=是）。可以是任意数字或整型标记名。

实例

StringInString("The mixer is running", "mix", 1, 0) 返回 5

StringInString("Today is Thursday", "day", 1, 0) 返回 3

StringInString("Today is Thursday", "day", 10, 0) 返回 15

StringInString("Today is Veteran's Day", "Day", 1, 1) 返回 20

StringInString("Today is Veteran's Day", "Night", 1, 1) 返回 0

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringLeft() 字符串

从文本最左端的字符开始，返回由 Chars 指定的字符数。

语法 MessageResult=**StringLeft**("*Text",Chars*);

参数	描述
Text	实际字符串或消息型标记名。
Chars	要返回的字符数目或整型标记名。

备注 如果 *Chars* 置为 0，则将返回整个字符串。

实例

StringLeft("The Control Pump is On", 3) 返回 "The"

StringLeft("Pump 01 is On", 4) 返回 "Pump"

StringLeft("Pump 01 is On", 96) 返回 "Pump 01 is On"

StringLeft("The Control Pump is On", 0) 返回 "The Control Pump is On"

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()**

StringLen() 字符串

返回文本长度为一整型值。

语法 IntegerResult=**StringLen**("*Text*");

参数	描述
Text	实际字符串或消息型标记名。

备注 将 *Text* 的长度（字符数）返回给 *IntegerTag*。消息型标记名中的所有字符，包括屏幕上通常不显示的特殊字符也计算在内。

实例

StringLen("Twelve percent") 返回 14

StringLen("12%") 返回 3

StringLen("The end."+ StringChar(13)) 返回 9

注意：回车键字符为 ASCII 13。

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()**

StringLower() 字符串

将文本中的所有大写字符转换为小写，并将结果字符串返回 MessageResult。

语法 MessageResult=**StringLower**("*Text*");

参数	描述
Text	将字符串转换成小写。实际字符串或消息型标记名。

备注 小写字符、符号、数字和其它特殊字符将不受影响。

实例

StringLower("TURBINE") 返回 "turbine"

StringLower("22.2 Is The Value") 返回 "22.2 is the value."

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()**

StringMid() 字符串

从位置 *StartChar* 开始，从文本中提取由 Chars 指定的字符数。此函数与其对应函数 **StringLeft()** 和 **ShringRight()** 稍有不同，它允许用户指定从消息标记中提取的字符串的首尾位置。

语法

```
MessageResult=StringMid("Text",StartChar,Chars);
```

参数	描述
Text	实际字符串或消息型标记名。
StartChar	指定要提取的第一个字符的位置。可以是任意数字或整型标记名。
Chars	指定要返回的字符总数。可以是任意数字或整型标记名。

实例

StringMid("The Furnace is Overheating",16,50) 返回 "Furnace"

StringMid("The Furnace is Overheating",13,3) 返回 "is "

StringMid("The Furnace is Overheating",16,50) 返回 "Overheating"

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringReplace() 字符串

替换或更改所给字符串的特定部分。使用此函数可用字符串标记名替换字符、单词或短语。

语法

```
MessageResult=StringReplace(Text,SearchFor,ReplaceWith,
CaseSens,NumToReplace,MatchWholeWords);
```

参数	描述
Text	您要更改的字符串。实际字符串或消息型标记名。
SearchFor	您要查找并替换的字符串。实际字符串或消息型标记名。
ReplaceWith	替换字符串。实际字符串或消息型标记名。
CaseSens	决定查找是否区分大小写（0=否，1=是）；数字或整型标记名。
NumToReplace	决定替换次数（-1=全部替换）；任意数字或整型标记名。
MatchWholeWords	决定函数是否限制为替换整个单词（0=否，1=是）；数字或整型标记名。 如果 *MatchWholeWords* 开启（置为 1），*SearchFor* 为“and”，则“handle”中的“and”将不会被替换。如果 *MatchWholeWords* 关闭（置为 0），则它会被替换。

实例

StringReplace("In From Within","In","Out",0,1,0) 返回 "Out From Within"（只替换第一处）

StringReplace("In From Within","In","Out",0,-1,0) 返回"Out From without"（全部替换）

StringReplace("In From Within","In","Out",1,-1,0) 返回"Out From Within"（替换所有匹配处）

StringReplace("In From Within","In","Out",0,-1,1) 返回"Out From Within" （替换所有的整个单词）

注意：StringReplace() 函数不识别特殊字符，例如 @#$%&*()。函数将其视为分隔符。例如，若函数 **StringReplace()** (abc#,abc#,1234,0,1,1)被执行，将不发生替换。“#”号被视为分隔符，而非字符。

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringRight() 字符串

从文本的最右边的字符开始，返回由 Chars 指定的字符数目。

语法

```
MessageResult=StringRight("Text",Chars);
```

参数	描述
Text	实际字符串或消息型标记名。
Chars	要返回的字符数或整型标记名。

备注

如果 *Chars* 置为 0，则将返回整个字符串。

实例

StringRight("The Pump is On", 2) 返回 "On"

StringRight("The Pump is On", 5) 返回 "is On"

StringRight("The Pump is On", 87) 返回 "The Pump is On"

StringRight("The Pump is On", 0) 返回 "The Pump is On"

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringSpace() 字符串

产生消息型标记名或表达式中的空格字符串。

语法 `MessageResult=`**`StringSpace`**`(`*`NumSpaces`*`);`

参数	描述
NumSpaces	要返回的空格数。可以是任意数字或整型标记名。

备注 **StringSpace()** 函数返回一个由 *NumSpaces* 指定长度的空格串。

实例 所有空格用“X” 字符表示：

`StringSpace(4)` 返回 `"××××"`

`"Pump" + StringSpace(1) + "Station"` 将返回 `"Pump×Station"`

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()**

StringTest() 字符串

测试文本的第一个字符来决定其是否为某一类型。

语法

`DiscreteResult=StringTest("Text",TestType)`

参数	描述
Text	函数将作用到的字符串。实际字符串或消息型标记名。
TestType	决定下列类型之一：

类型	描述
1	字母数字符号（'A'-'Z', 'a-z' 和 '0-9'）
2	数字符号 ('0'- 9')
3	字母字符（'A-Z' 和 'a-z'）
4	大写字母 ('A'-'Z')
5	小写字母 ('a'-'z')
6	标点符号 (0x21-0x2F)
7	ASCII 字符 (0x00 - 0x7F)
8	十六进制字符（'A'-'F' 或 'a'-'f' 或 '0'-'9'）
9	可打印字符 (0x20-0x7E)
10	控制字符（0x00-0x1F 或 0x7F）
11	空白符（0x09-0x0D 或 0x20）

备注

如果 *Text* 中的首字符是 *TestType* 指定的类型，则 **StringText()** 函数会向返回一个正值给 *DiscreteResult*。如果 **StringTest()** 函数包含一个以上的字符，则只会测试标记名的第一个字符。

实例

StringTest("ACB123",1) 返回 1

StringTest("ABC123",5) 返回 0

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

StringToIntg() 字符串

将消息型标记名的数字值转换成一个能运用数学运算的整型数值。

语法

```
IntegerResult=StringToIntg("Text");
```

参数	描述
Text	函数将作用到的字符串。实际字符串或消息型标记名。

备注

当对此语句求值时，系统将读取字符串首字符的数字值。如果首字符不是一个数字（空格将被忽略），则字符串的值等于零 (0)。如果首字符是一个数字，则系统继续读取后续字符，直至遇到一个非数字值为止。

实例

```
If Text="ABCD", then IntegerTag=0.
If Text="22.2 is the Value", then IntegerTag=22 (since integers are whole numbers).
If Text="The Value is 22", then IntegerTag=0.
```

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToReal(), StringTrim(), StringUpper(), Text()

StringToReal() 字符串

将消息型标记名的数值转换成可应用数学运算的实型（浮点）值。

语法

`RealResult=`**`StringToReal`**`("`*`Text`*`");`

参数	描述
Text	函数将作用到的字符串。实际字符串或消息型标记名。

备注

当对此语句求值时，系统将读取字符串首字符的数字值。如果首字符不是一个数字（空格将被忽略），则字符串的值等于零 (0)。如果首字符是一个数字，则系统继续读取后续字符，直至遇到一个非数字值为止。

实例

If *Text*="ABCD", then *RealTag*=0.

If *Text*="22.261 is the Value", then *RealTag*=22.261.

If *Text*="The Value is 22", then *RealTag*=0.

参见

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringTrim(), StringUpper(), Text()

StringTrim() 字符串

从文本中移除不需要的空格。

语法 `MessageResult=StringTrim("Text",TrimType);`

参数	描述
Text	要删除空格的字符串。实际字符串或消息型标记名。
TrimType	决定下列类型之一：

类型	描述
1	删除前导空格（第一个非空格字符的左边）
2	删除尾部空格（最后一个非空格字符的右边）
3	删除单词间单个空格外的多余空格

备注 查找 *Text* 中要移去的空格（ASCII 0x9-0x0D 或 0x20）。*TrimType* 决定函数的使用方式：

实例 所有空格用“X” 字符表示：

StringTrim("×××××This×is×a××test×××××", 1) 返回 "This×is×a××test×××××"

StringTrim("×××××This×is×a××test×××××", 2) 返回 "×××××This×is×a××test"

StringTrim("×××××This×is×a××test×××××", 3) 返回 "This×is×a×test"

注意：StringReplace() 函数可用于从指定的消息型标记名中删除全部空格。只需用“空 (null)”替换全部空格就可以。

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringUpper(), Text()**

StringUpper() 字符串

将文本中的全部小写转换成大写。

语法 `MessageResult=StringUpper("Text");`

参数	描述
Text	要转换为大写的字符串。实际字符串或消息型标记名。

备注 大写字符、符号、数字以及其它特殊字符将不受影响。

实例 **StringUpper("abcd") 返回"ABCD."**

StringUpper("22.2 is the value") 返回 "22.2 IS THE VALUE"

参见 **StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringFromTimeLocal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), Text()**

Tan() 数学

返回给定角（以度表示）的*正切*值。

语法 `Result=Tan(AngleNumber);`

参数	描述
AngleNumber	角的度数。任意数字、实型或整型标记名。

实例 **`Wave = 10 + 50 * Tan(6 * $Second);`**

`Tan(45) 返回 1`

`Tan(0) 返回 0`

参见 **Cos(), Sin(), ArcCos(), ArcSin(), ArcTan()**

Text() 字符串

使消息型标记名根据指定的 *Format_Text* 显示模拟（整型或实型）标记名的值。

语法 MessageResult=**Text**(Analog_Tag,"Format_Text");

参数	描述
Analog_Tag	任意数字、实型或整型标记名。
Format_Text	转换中使用的格式。实际字符串或消息型标记名。

实例

```
MessageTag=Text(Analog_Tag,"#.00");
```

MessageTag 是消息型标记名，*66* 是一个整型或实型标记名，“#0.00”表示等价值的显示格式：

```
If Analog_Tag=66, then MessageTag=66.00.
If Analog_Tag=22.269, then MessageTag=22.27.
If Analog_Tag=9.999, then MessageTag=10.00.
LogMessage("The current value of FreezerRoomTemp is:"+ Text
(FreezerRoomTemp, "#.#"));
```

在下面的例子中，MessageTag 将置为“One=1 Two=2”。

```
MessageTag = "One + " + Text(1,"#") + StringChar(32) +
"Two +" + Text(2,"#");
```

参见 **StringFromIntg(), StringToIntg(), StringFromReal(), StringToReal()**

Trunc() 数学

通过简单地删除小数点右边的部分来截短一个实型（浮点）数字。

语法 ResultNumericTag=**Trunc**(*Number*);

参数	描述
Number	任意数字、实型或整型标记名。

备注 此函数与将实型标记名内容放入整型标记名中效果相同。

实例

```
Trunc(4.3) 返回 4
Trunc(-4.3) 返回 -4
```

参见 **Round()**

TseGetClientId() 终端服务

如果 View 应用程序当前在终端服务客户端上运行，此函数返回字符串版本的客户端 ID（客户端的 TCP/IP 地址），否则返回一个空的字符串。

语法

```
MessageResult=TseGetClientId();
```

参数	描述
MessageResult	显示客户端 ID 的消息型标记名。

备注

此 ID 供内部使用，用于生成 SuiteLink 服务器名及 logger 文件名。

实例

MsgTag=TseGetClientID();

客户端 IP 地址，例如 **10.103.202.1** 将返回给 MsgTag。

TseQueryRunningOnConsole() 终端服务

如果 View 应用程序当前在终端服务器控制台运行，此函数将返回一个非零的整型值，否则返回零 (0) 值。

语法

```
Result=TseQueryRunningOnConsole();
```

参数	描述
Result	如果 View 没有在 TS 控制台上运行，返回 0 。

备注

无

实例

IntTag=TseQueryRunningOnConsole();

如果终端服务控制台上在运行 View，返回 IntTag=1。

TseQueryRunningOnClient() 终端服务

如果 View 应用程序当前在终端服务器客户服务端运行，此函数将返回一个非零的整型值，否则返回零 (0) 值。

语法

```
Result=TseQueryRunningOnClient();
```

参数	描述
Result	如果 View 没有在 TS 客户端运行，返回 0 。

备注

无

实例

IntTag=TseQueryRunningOnClient();

如果终端服务客户端上在运行 View，返回 IntTag=1。

wcAddItem() 窗口控件

将所给字符串添加到列表框或组合框。如果列表框或组合框在创建时未排序，则字符串将添加到列表的结尾。否则字符串将插入到列表中，并对列表重新排序。

语法

```
[ErrorNumber=]wcAddItem("ControlName","MessageTag");
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
MessageTag	要显示的消息型字符串；实际字符串或消息型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当打开包含列表框的窗口（使用“显示时”窗口 QuickScript）时，下面的语句会将消息型字符串的内容添加到列表框中：

```
wcAddItem("ListBox_1", "Chocolate");
wcAddItem("ListBox_1", "Vanilla");
wcAddItem("ListBox_1", "Strawberry");
```

参见

wcInsertItem()

wcClear() 窗口控件

从列表框或组合框中删除所有项目。

语法 [ErrorNumber=]**wcClear**(*"ControlName"*);

参数	描述
ControlName	窗口控件对象名；例如 ListBox_1；实际字符串或消息型标记名。

备注 有关所返回错误号的列表，请参阅附录 A。

应用于 列表框和组合框。

实例 当执行触动按钮动作 QuickScript 时，下面的语句将清除组合框中的所有项目：

```
wcClear("ListBox_1");
```

wcDeleteItem() 窗口控件

从列表框或组合框中删除与项目索引参数有关的项。

语法 [ErrorNumber=]**wcDeleteItem**(*"ControlName",ItemIndex*);

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
ItemIndex	与项目位置对应的一个数字。可以是任意数字或整型标记名。

备注 有关所返回错误号的列表，请参阅附录 A。

应用于 列表框和组合框。

实例 当执行触动按钮动作 QuickScript 时，下面的语句将删除列表中的第三个项目：

```
wcDeleteItem("ListBox_1", 3);
```

wcDeleteSelection() 窗口控件

从列表中删除当前选定的项目。

语法

[ErrorNumber =]**wcDeleteSelection**(*"ControlName"*);

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当执行触动按钮动作 QuickScript 时，下面的语句将删除组合框中当前选定的项目：

```
wcDeleteSelection("ListBox_1");
```

wcErrorMessage() 窗口控件

返回一个描述错误的消息型字符串。

语法

ErrorMessage=**wcErrorMessage**(*ErrorNumber*);

参数	描述
ErrorMessage	消息型标记名。
ErrorNumber	由所有窗口控件函数返回的数字。可以是任意数字或整型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框、文本框、组合框、复选框和单选钮。

实例

如果在加载列表时发生错误，则在消息标记名 *ErrorDescription* 中显示描述错误的文本。在本例中，通过对标记名 *ErrorDescription* 指定字符串值输出动画链接来显示错误消息。

“显示时”窗口 QuickScript：

```
ErrorNumber=wcLoadList("ListBox_1","c:\InTouch\recipe.txt")
;
ErrorDescription=wcErrorMessage(errornumber);
```

此函数也在所有窗口控件函数中用于显示错误消息：

```
ErrorNumber=wcAddItem("ListBox_1","United States");
ErrorMsg=wcErrorMessage(ErrorNumber);
```

wcFindItem() 窗口控件

确定列表框或组合框中与 *Message* 字符串相匹配的第一个项所对应的索引。

语法

```
[ErrorNumber=]wcFindItem
("ControlName","MessageTag",DiscreteTag,Tagname);
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
MessageTag	要比较的消息型字符串。实际字符串或消息型标记名。
DiscreteTag	决定字符串比较的类型。可赋予下列离散值之一： 0 = 不区分大小写 1 = 区分大小写
Tagname	整型标记名的实际名称。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框、文本框、组合框、复选框和单选钮。

实例

如果在加载列表时发生错误，则在消息标记名 *ErrorDescription* 中显示描述错误的文本。在本例中，通过对标记名 *ErrorDescription* 指定字符串值输出动画链接来显示错误消息。

wcGetItem() 窗口控件

返回与列表框或组合框中相应的 *ItemIndex* 相关联的项目字符串的值属性。

语法

`[ErrorNumber=]`**`wcGetItem`**`("ControlName",ItemIndex,Tagname);`

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
ItemIndex	与项目位置对应的一个数字。可以是任意数字或整型标记名。
Tagname	实型或整型标记名的实际名称。**wcGetItem** 函数在函数返回时，会将与此项目对应的数值放入此标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当执行触动按钮动作 QuickScript 时，下面的语句会将组合框中第十个项目的字符串值返回给消息标记名 *ListSelection*：

```
wcGetItem("Combobox_1", 10, ListSelection);
```

如果列表中的第十项是“Vanilla”，则 *ListSelection* 将包含字符串 Vanilla。

wcGetItemData() 窗口控件

确定与参数 ItemIndex 所确定的列表项关联的整数值。

语法

```
[ErrorNumber=]wcGetItemData("ControlName",ItemIndex,
Tagname);
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
ItemIndex	与项目位置对应的一个数字。可以是任意数字或整型标记名。
Tagname	实型或整型标记名的实际名称。**wcGetItemData** 函数在函数返回时，会将与项目对应的数值放入此标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当执行触动按钮动作 QuickScript 时，下面的语句将检索与列表框中第五项关联的数字值，并将其返回给标记名 *ItemValue*：

```
wcGetItemData("ListBox_1", 5, ItemValue);
```

如果列表中的第五项被赋予整数值 4500，则标记名 *ItemValue* 将包含 4500。

参见

wcSetItemData()

wcInsertItem() 窗口控件

在列表中插入*消息型*字符串。*ItemIndex* 是列表中要插入字符串位置的相应索引号。与 **wcAddItem()** 不同，**wcInsertItem()** 函数不会对列表排序，即使它是一个已排序的列表框或组合框。

语法

```
[ErrorNumber=]wcInsertItem("ControlName",ItemIndex,
"MessageTag");
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
ItemIndex	要添加项目的位置所对应的编号。如果此参数为 -1，则字符串将添加到列表结尾。可以是任意数字或整型标记名。
MessageTag	包含插入 *ItemIndex* 指定位置的字符串。实际字符串或消息型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当执行触动按钮动作 QuickScript 时，下面的语句将一个名为“Blueberry”的新项目插入到列表框从上往下数的第 4 个位置处。

```
wcInsertItem("ListBox_1", 4, "Blueberry");
```

参见

wcAddItem()

wcLoadList() 窗口控件

用 *FileName* 中包含的项目替换列表框或组合框的内容。

语法

[ErrorNumber=]**wcLoadList**(*"ControlName","Filename"*);

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
Filename	包含纯 ASCII 格式的文件名。如果参数没有提供完整的路径名，则此函数将在应用程序目录中查找消息文件。实际字符串或消息型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当打开包含组合框的窗口（“显示时”QuickScript）时，下面的语句会将一个适当格式的列表（位于 C：\InTouch\recipe.txt）加载到此组合框中：

```
wcLoadList("Combobox_1", "c:\InTouch.32\wclist.txt");
```

参见

wcAddItem(), wcSaveList()

wcLoadText() 窗口控件

用 *FileName* 的内容替换文本框的内容。

语法

```
[ErrorNumber=]wcLoadText("ControlName", "Filename");
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
Filename	包含纯 ASCII 格式的文件名。如果参数没有提供完整的路径名，则此函数将在应用程序目录中查找消息文件。实际字符串或消息型标记名。

备注

wcLoadText() 函数只支持纯 ASCII 文本格式文件，例如些用 Microsoft 记事本程序创建的文件。

有关所返回错误号的列表，请参阅附录 A。

有关高级文件浏览功能，请参阅 Factory Suite 的 *Productivity Pack* 的文档浏览器。

应用于

文本框。

实例

当打开包含文本框的窗口（“显示时”QuickScript）时，下面的语句会将记事本文本文件 (c:\InTouch.32\readme.txt.) 加载到文本框中：

```
wcLoadText("Textbox_1", "c:\InTouch.32\readme.txt");
```

wcSaveList() 窗口控件

用列表框或组合框对象中的项目替换 *FileName* 的内容。

语法

```
[ErrorNumber=]wcSaveList("ControlName","Filename");
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
Filename	包含纯 ASCII 格式的文件名。如果文件不存在，则会创建一个。项目通过 **wcLoadList()** 函数陆续加载到列表对象中。实际字符串或消息型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

列表框和组合框。

实例

当执行触按钮动作 QuickScript 时，下面的语句将列表框中的当前项目保存到文件中 **(c:\InTouch\newList.txt)**：

```
wcSaveList("ListBox_1", "c:\InTouch.32\newlist.txt");
```

如果使用外部 ASCII 文件来填充列表和组合框，则必须遵照特定的格式并且包含特定的信息。格式如下所示：

ControlType, ListCount
ListItem, ItemData
ListItem, ItemData
: :
: :
ListItem, ItemData

例如：将一个列表文件加载到组合框中，它包含三个可选项，并且这些项目没有赋予项目数据（有关项目数据的详情，参见 **wcSetItemData()** 函数）。文件的格式将如下：

COMBOBOX, 3
Chocolate, 0
Vanilla, 0
Strawberry, 0

说明：COMBOBOX 是控件类型。**ListCount** 为 3，表示 Chocolate、Vanilla 和 Strarberry 三个项目。Chocolate 将被列为第一个列表项或索引 1；Vanilla 列为索引 2；Strawberry 列为 3。每个索引项目均具有数据值 0。

参见

wcLoadList(), wcSetItemData()

wcSaveText() 窗口控件

将包含于文本框中的文本存储于 *FileName*。如果此文件不存在，则会创建一个。如果已经存在，则必须是可读/写的。

语法

[ErrorNumber=]**wcSaveText**(*"ControlName","Filename"*);

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
Filename	包含纯 ASCII 格式的文件名。如果没有提供完整的路径名，则此函数将保存在应用程序目录下。如果文件不存在，则会创建一个。项目通过 **wcLoadList()** 函数陆续加载到列表对象中。实际字符串或消息型标记名。

备注

有关所返回错误号的列表，请参阅附录 A。

应用于

文本框。

实例

当执行触按钮动作 QuickScript 时，下面的语句将文本框中当前输入的信息保存到文件 c:\InTouch.32\newtext.txt 中：

```
wcSaveText("Textbox_1", "c:\InTouch.32\newtext.txt");
```

注意：wcSaveText() 函数仅以纯 ASCII 文本文件格式保存文件，例如用 Microsoft 记事本创建的文件。

参见

wcLoadList()

wcSetItemData() 窗口控件

将项目的整数值 (*Number*) 赋予由 *ItemIndex* 指定的列表中的项目。此函数允许对字符串赋予一个数字。

语法

```
[ErrorNumber=]wcSetItemData("ControlName",ItemIndex,
Number);
```

参数	描述
ControlName	窗口控件对象名，例如 ListBox_1；实际字符串或消息型标记名。
ItemIndex	指定请求项目的显示顺序的整数值。可以是任意数字或整型标记名。
Number	表示项目数据的任意整型值。可以是任意数字或整型标记名。

备注

可以在外部（如使用记事本）创建一个包含项目的完整列表，然后通过函数调用进行加载。列表必须如 **wcSaveList()** 函数所述正确格式化。

有关所返回错误号的列表，请参阅附录 A。

实例

某个配方包含三种成分：面粉、糖和盐。其中面粉 4500 克，糖 1500 克，盐 325 克。通过数据改变 QuickScript 将这些值分配给列表框中的每一项，该数据改变 QuickScript 由所选配方 (Tagname, RecipeName) 触发：

```
wcSetItemData("ListBox_1", 1, 4500);   {设定列表中的第 1 项
(flour)=4500}
wcSetItemData("ListBox_1", 2, 1500);   {设定列表中的第 2 项
(sugar)=1500}
wcSetItemData("ListBox_1", 3, 325);   {设定列表中的第 3 项
(salt)=325}
```

wcGetItemData() 函数用于返回与列表项目相关联的值（项目数据）。*Tagname* 参数包含返回的数值。此参数可以是一个直接写入真实设备的 I/O 整型标记名。

参见

wcLoadList(), wcSaveList(), wcGetItemData()

WWControl() 其它

允许您从 InTouch 恢复、最小化、最大化或关闭应用程序。

语法

```
WWControl("AppTitle","ControlType");
```

参数	描述
AppTitle	要控制的应用程序标题的名称。特定应用程序的标题可以通过 **InfoAppTitle()** 函数确定。实际字符串或消息型标记名。
ControlType	决定以何种方式控制应用程序如下 (这些动作等价于单击此应用程序控制菜单中它们所对应选项)。实际字符串或消息型标记名。

类型	描述
"Restore"	激活并显示应用程序窗口。
"Minimize"	激活窗口并显示为一个图标。
"Maximize"	激活并显示应用程序窗口。
"Close"	关闭应用程序。

实例

```
WWControl("Calculator","Restore");
WWControl(InfoAppTitle("View"), "Close");
```

参见

InFoAppTitle(), ActivateApp(), StartApp()

WWExecute() WWDDE

对指定的*应用程序*和*主题*执行命令（使用 DDE 协议）。

语法

```
[Status=]WWExecute("Application","Topic","Command");
```

参数	描述
Application	执行命令所发送到的应用程序。实际字符串或消息型标记名。
Topic	执行命令所发送到的应用程序中的主题；实际字符串或消息型标记名。
Command	要发送的命令；实际字符串或消息型标记名。

备注

将 *Command* 字符串发送到指定的*应用程序*和*主题*。

实例

下面的指令执行 Excel 中的一个宏：

```
Macro="Macro1!TestMacro";
Command="[Run(" + StringChar(34) + Macro + StringChar(34)
 + ",0)]";
WWExecute("excel","system",Command);
```

当执行 WWExecute("excel","system",Command）时，下面的命令将发送给 Excel (而且 *TestMacro* 将运行)：

```
[Run("Macro1!TestMacro")]
```

下面的 QuickScript 执行 Microsoft Access 中的一个宏：

```
WWExecute("MSAccess","system","MyMacro");
```

如果应用程序正在运行、主题存在，并且命令发送成功，则 **WWExecute** 函数会返回 1。如果应用程序繁忙，它将返回 0；如果出错，则返回 -1。因此，可监视此命令的状态：如果应用程序正在运行、主题存在，并且命令发送成功，则 **WWExecute** 函数会返回 1。如果应用程序繁忙，它将返回 0；如果出错，则返回 -1。因此，可监视此命令的状态：

```
Status=WWExecute("excel","system",Command);
```

Status 是一个为 1、-1 或 0 的整型标记名。

WWPoke() WWDDE

使用 DDE 协议将值插入指定的*应用程序*、*主题*和*项目*。

语法

```
[Status=]WWPoke("Application","Topic","Item","TextValue");
```

参数	描述
Application	插入命令所发送到的应用程序；实际字符串或消息型标记名。
Topic	插入命令所发送到应用程序内的主题；实际字符串或消息型标记名。
Item	主题中要插入的项目；实际字符串或消息型标记名。
TextValue	消息变量或字符串。如果要发送的值是一个数字，您可以使用 **Text()**、**tringFromIntg()** 或 **StringFromReal()** 函数对其进行转换；实际字符串或消息型标记名。

备注

在此语句中，*TextValue* 的值被发送到指定的*应用程序*、*主题*和*项目*。

实例

下面的语句将数值转换为文本并且将它插入 Excel 电子表格的单元格中：

```
String=Text(Value,"0");
WWPoke("excel","[Book1.xls]sheet1","r1c1",String);
```

有关在 InTouch (DDE) 中使用 Excel 5.0 的详细信息，请参阅《*InTouch 用户指南*》。

注意：WWPoke() 从应用程序 "View" 到 "View" 的行为没有定义并且不被支持。不能保证 WWPoke() 命令在这种情况下会成功，该命令可能在没有得到期望结果时超时。

如果应用程序正在进行、主题和项目存在，并且值成功发送，则 **WWPoke()** 函数会返回 1。如果应用程序繁忙，它将返回 0；如果出错，则返回 -1。因此，可监视此命令的状态：

```
Status=WWPoke("excel","[Book1.xls]sheet1","r1c1",String);
```

Status 是一个为 1、-1 或 0 的整型标记名。

参见

Text(), StringFromIntg(), StringFromReal()

WWRequest() WWDDE

对特定*应用程序*、*主题*和*项目*发出一次性取值的请求（使用 DDE 协议）。

语法

WWRequest(Application,Topic,Item,ValueMsg_Tag);

参数	描述
Application	从中请求数据的应用程序。实际字符串或消息型标记名。
Topic	从中请求数据的应用程序中的主题；实际字符串或消息型标记名。
Item	从中请求数据的主题内的项目；实际字符串或消息型标记名。
ValueMsg_Tag	以引号括起的消息型标记名，包含来自应用程序、主题和项目的请求值；实际字符串或消息型标记名。

备注

在此语句中，特定*应用程序*、*主题*和*项目*中的 DDE 值将返回给 *ValueMsg_Tag*。

值将*作为字符串*返回给消息型标记。如果该值为一个数字，您可以使用 **StringToIntg()** 或 **StringToReal()** 函数对其进行转换。

实例

下面的语句从 Excel 电子表格单元格中请求一个值，并将所得字符串转换成数值：

```
WWRequest("excel","[Book1.xls]sheet1","r1c1",Result);
Value=StringToReal(Result);
```

如果应用程序正在运行、主题和项目存在，并且数值成功返回，则 **WWRequest()** 将返回 1。如果应用程序繁忙，它将返回 0；如果出错，则返回 -1。因此，可监视此命令的状态：

```
Status=WWRequest("excel","[Book1.xls]sheet1","r1c1",Result)
;
```

Status 是一个为 1、-1 或 0 的整型标记名。

参见

StringToIntg(), StringToReal()

第 4 章

OLE 自动化与 InTouch

Microsoft 对象链接与嵌入 (OLE) 技术支持客户端与服务器之间的各种信息交换和处理机制。InTouch 可以充当许多此类服务器的客户端。InTouch 应用程序可用于在标记名值改变时通知服务器，允许服务器执行通过编程指定的任何动作。此功能可以扩展 InTouch 的特性，使之超越其自身内置的能力。

本章概述 OLE 自动化结构，同时介绍通过这些 InTouch 扩展功能来实现 OLE 自动化的步骤。由于 InTrack 是一个 OLE 自动化服务器，因此本章的许多实例中使用它来向您进行介绍。

OLE 自动化基础

OLE 以极其简便的方式，提供了客户端与服务器之间的标准通讯机制。

OLE 的自动化对象元件提供了客户端/服务器环境下的面向对象的编程机制。服务器通过对象类提供特定的计算技术。客户端则通过创建对象并访问其成员来确保使用这些技术。OLE 提供客户端和服务器之间的通讯连接。

为理解自动化对象，必须对面向对象的编程概念有一个基本了解。

- *对象*是根据对象类创建的可识别独立实体。
- 类定义了相关的一组*属性*（数据）和*方法*（函数）。因此，相同类的所有对象具有相同的属性和方法集，虽然每个对象可能具有不同的属性值。

InTouch OLE 自动化客户端扩展

InTouch 扩展使用三个主要元件提供对 InTrack 自动化对象的简便访问：

- InTouch 中添加了八个附加的内置函数。

  OLE_CreateObject OLE_IsObjectValid
  OLE_GetLastObjectError OLE_ReleaseObject
  OLE_GetLastObjectErrorMessage OLE_ResetObjectError
  OLE_IncrementOnObjectError OLE_ShowMessageOnObjectError

- 一元操作符指示 OLE 自动化表达式的开始。
- InTrack OLE 浏览器提供所有 InTrack 自动化对象类的模板和联机帮助。通过“脚本编辑器”对话框右下角的“InTrack OLE”按钮，可访问该浏览器。

InTouch 自动化表达式

InTouch 内的所有自动化对象引用均由一个百分号开始，后跟标识对象引用的名称。该名称必须由字母字符开头，可包括字母、数字和下划线字符。名称区分大小写并且具有全局性。不同对象名的例子如下：

- %ObjectName（或 %objectName 或 %objectname）
- %Lot5
- %lot_5
- %aMachine
- %A_Machine
- %Sublot

不要将对象名与 InTouch 标记名混淆，但 OLE 自动化对象可与 InTouch 标记同名（不同的是前面加了一个百分号）。

对象引用

上述“百分号名称”也称为*对象引用*，因为它们充当自动化对象的指针。使用下面的 InTouch 脚本语句，两个不同的名称可以访问相同的对象：

```
%Object1 = %Object2;
```

在执行此语句后，两个对象引用均“指向”相同的自动化对象。

对象引用名称在运行期间首次使用时自动定义。它们具有全局性，也就是说，您可以在一个脚本中对其初始化，然后将其用于另一脚本中。

大多数面向对象的编程语言都支持使用“点”操作符来引用对象的属性和方法。下面的实例显示使用点操作符将对象属性值设置为 5：

```
%Object.Property = 5;
```

嵌套对象（点操作符）

如果一个对象是其它对象的成员，则可以多次使用“点”操作符。下面将嵌套对象的属性值设置为 6：

```
%Object.Property_For_Nested_Object.Property = 6;
```

使用点操作符时需要遵循一些特定的规则：

1. 对象引用规格以及任意成员引用内不得嵌有空格。

- 百分号和对象引用名称开头之间不得包含空格。
- 对象引用名、“点”操作符和类成员名之间不得空格。
- 嵌套对象的属性名、“点”操作符与嵌套对象的属性名之间不得有空格。

2. 从百分号（包括百分号）开始到最后一个成员引用，整个规格的字符数不得超过 98。
   例如，规格 %Object.Property 的长度为 16 个字符，规格 %Object.Property_For_Nested_Object.Property 的长度为 44 个字符。

InTouch 到 OLE 的数据类型转换

不是所有的 InTouch 数据类型均受 OLE 扩展支持。下面概括介绍支持的数据类型和数据转换。注意转换过程在*运行时*而不是在 WindowMaker 中进行。因此，数据转换在意外结果时可能不会报告错误。

整型 32 位有符号值对象引用可转换为下列数据类型：

实型	数值为浮点型。
消息型	到一系列小数位的标准转换 – 可能前面跟负号。
离散型	如果值非零，转换为真 (1)，否则转换为假 (0)。

实型 64 位解析度。InTouch 使用 64 位解析度进行计算，但将结果存储在 32 位解析度标记中。自动化对象引用支持更高解析度。对象引用可转换为下列数据类型：

整型	实数值将被截断。
消息型	到一系列小数位的标准转换，小数点前面可能跟负号。
离散型	如果值非零，转换为真 (1)，否则转换为假 (0)。

消息型	限于 131 个字符的字符串（InTouch 的约束条件）。此数据类型的数据转换更加复杂，并很可能导致出现运行时错误。	
	整型	与 InTouch 内置函数 **StringToIntg** 相同。如果字符串以数字开头（不包括空格但包括负号），将转换该数字。任何其它值均转换为零。
	实型	与 InTouch 内置函数 **StringToReal** 相同。如果字符串以数字开头（不包括空格但包括负号），将转换该数字。任何其它值均转换为零。
	离散型	转换整数，然后运用规则将整型转换为离散型。
	对象引用	此转换仅在极少数特殊场合下需要，并且应尽量避免使用。滥用此转换机制可能导致 WindowViewer 因致命错误而退出。如果输入值不是由对象引用转换所产生的字符串，此转换会报告错误。
离散型	对象引用可转换为下列数据类型：	
	整型	真为 1，否则为 0。
	实型	真为 1，否则为 0。
	消息型	真转换为 “0”，假转换为 “-1”。
对象引用	通常，对象引用不得用于可能发生数据转换的环境中。对象引用可转换为下列数据类型：	
	消息型	特殊格式的字符串，包括对象引用的十六进制表示。

限制

InTouch 自动化对象引用有两条重要的限制条件：

- 扩展仅限于在 InTouch 脚本内进行。
- 扩展使用“延迟绑定”— 在脚本执行之前，大多数的错误均无法识别。

创建对象：OLE_CreateObject

自动化对象通过 InTouch 内置系统函数 OLE_CreateObject 进行创建，此函数的语法如下：

```
OLE_CreateObject(%Object, “className”);
```

此处：

OLE_CreateObject	用于创建自动化对象的函数。
%Object	用于引用自动化对象的名称。
className	由 OLE 服务器提供的类名。

此函数不返回值。

实例

例如，要创建 InTrack 类 Sublot 的对象，应使用下面的语句：

```
OLE_CreateObject(%Sublot, “InTrack.Sublot”);
```

“类名”拼写错误是一个常见错误，它无法在 WindowMaker 内识别。在 WindowViewer 中执行函数之前，此错误将不能识别和报告。为避免此拼写错误，请使用 InTrack OLE 浏览器中的 OLE_CreateObject 模板。

当首次创建对象时，服务器会将其属性自动设为适当的缺省值。

注意：对象引用名不需要通过调用 OLE_CreateObject 来初始化。在上面的例子中，使用赋值语句将对象引用与对象关联。

管理对象：OLE_ReleaseObject

只要至少有一个对象对之引用，则该对象即属有效。但是，最好释放那些不再使用的对象，以节省系统资源。要释放对象：

1. 用相同的对象引用名创建一个新对象。

   当使用与现有自动化对象引用相同的名称创建新的自动化对象时，原始自动化对象将被释放，新对象将通过原始名称关联。例如：

   ```
   OLE_CreateObject(%Object, "InTrack.Sublot");
   OLE_CreateObject(%Object, "InTrack.DateTime");
   ```

   第一条语句让 %Object 创建和引用一个 Sublot 类的对象。第二条语句释放 Sublot 类的对象，然后创建一个 DateTime 类的对象并将其与 %Object 关联。

2. 使用内置的 OLE_ReleaseObject 系统函数。此函数的语法如下：

   ```
   OLE_ReleaseObject(%objectName);
   ```

   此处：

%objectName	自动化对象引用名称

   此函数不会返回值，并且始终能够顺利执行。

3. 设置与第二个对象的引用。

   将某一自动化对象设为另一自动化对象的语法为：

   ```
   %object1 = %object2;
   ```

   此处：

%object1	是“指向”与 %object2 相同对象的对象引用。先前与 %object1 关联的自动化对象将被释放。
%object2	通过名称 %object1 引用的自动化对象。此对象必须为相同类型。

   在本例中，任何与 %object1 关联的对象均将释放。

测试有效对象： OLE_IsObjectValid

在通过 OLE_ReleaseObject 函数中断对象引用与 OLE 自动化对象之间的关系后，引用将“无效”。在任何对象表达式中使用无效引用将导致运行时错误。

InTouch 内置系统函数 OLE_IsObjectValid 可启用对象引用的有效性检查。此函数的语法如下：

```
OLE_IsObjectValid(%objectName);
```

此处：

%objectName	要从内存释放的自动化对象引用名称。

如果 %objectName 当前连接至有效的自动化对象，此函数将返回离散（布尔）值。

获得属性值

属性是定义对象特征或其行为方面的既定内容。点操作符将对象引用与属性名分隔。当 InTouch 脚本允许使用标记时，可使用下面的语法来返回自动化对象属性的值：

```
%Object.Property
```

此处：

%Object	自动化对象引用
Property	对象属性

如果对象属性为对象自身，则可以多次使用“点”操作符，如下例所示：

```
%Object.Property1.Property2
```

此处：

%Object	自动化对象引用
Property1	%Object 的属性。注意属性可以是嵌套对象，也可以是与另一对象的引用。InTouch 和 OLE 会自动处理这两种情况。
Property2	从中检索值的嵌套对象的属性。

使用多个点转换成多个“取属性”操作。在本实例中，对 %Object.Property1 执行简单的“取”操作，该对象根据 OLE 自动化标准，返回与嵌套对象的引用。然后，使用返回的引用再对 Property2 执行简单的“取”操作。这可以通过下面的两个 InTouch 脚本语句手动执行：

```
%tempObject = %Object.Property1;
%tempObject.Property2;
```

因为以下几个方面的原因，这一系列动作很重要：

1. 每个点操作符都可能出现排错属性名。
2. 您可以在各种环境下使用多个点来简化编码过程，例如获取属性、设置优先级和调用方法等。

实例 1

本实例在时间为 1 号凌晨 1:00 到 2:00 之间递增 InTouch 标记值。此段代码阐述了几个概念：

```
OLE_CreateObject(%DateTime, "InTrack.DateTime");
%DateTime.SetLocalTime();
IF (%DateTime.Day == 1) AND (%DateTime.Hour > 1) AND (%DateTime.Hour < 2)
THEN
    Tag = Tag + 1;
ENDIF;
OLE_ReleaseObject(%DateTime);
```

实例 2

本实例说明如何多个“点”操作符。其目标是将机器资源的容量信息读入 InTouch 标记，以便显示给操作员。本实例是高度灵活的，阐述了在各种 InTouch 表达式环境中如何获取属性。

为简便起见，假定对象 %Machine 已创建为 InTrack.机器的一个实例，并且已从 InTrack 数据库加载适当的信息（此操作的机制将在后面的章节介绍）。

```
NumerUnitsTag = %Machine.Capacity.Quantity;
TypicalWeightTag = %Machine.Capacity.Quantity * 40.6;
ContrivedLoadingTag = Log(%Machine.Capacity.Quantity) + 1;
UnitsNameTag = %Machine.Capacity.Units;
```

设置属性

输入或输出对象的属性值可使用三种方法设置：

- 使用赋值语句（如下所述）。
- 将属性引用作为输出参数传递给方法。
- 调用方法以在单个动作中设置对象的一个或多个属性（新值作为参数传递）。这些机制与类有关。有关详情，请参阅下面章节的方法引用。

注意: 引用自动化对象的属性不能作为“被引用”（输出）参数从 InTouch 脚本传递给所调用的 DLL 函数。对象的属性值不能被 DLL 函数更改。

通过赋值语句设置对象属性值的基本语法如下所示：

```
%Object.Property = Value;
```

此处：

%Object	自动化对象引用
Property	对象属性
Value	您要赋给对象属性的值

在获取对象属性时，如果对象属性本身为对象，则可多次使用“点”操作符：

```
%Object.Property1.Property2 = Value;
```

此处：

%Object	自动化对象引用
Property1	同类对象 %Object 属性
Property2	具有值集的嵌套对象的属性

上面这个使用多个“点”操作符的 InTouch 语句相当于：

```
%tempObject = %Object.Property1;
%tempObject.Property2 = Value;
```

实例

本实例的第一行创建一个名为 %PrimaryAmt 的“Amount”类自动化对象。脚本的第二行和第三行分别将 %PrimaryAmt 自动化对象的“Amount”属性设为“100”、“Units”属性设为“each”。

```
OLE_CreateObject(%PrimaryAmt, “InTrack.Amount”);
%PrimaryAmt.Quantity = 100;
%PrimaryAmt.Units = “each”;
```

调用方法

调用自动化对象的成员方法是客户端要求服务器以其名义执行工作的常用途径。下面的语句介绍如何对自动化对象调用方法：

```
resultValue = %Object.Method(Parameter1, Parameter2, ...);
%Object.Method(Parameter1, Parameter2, ...);
```

此处：

resultValue	方法返回的数据值
%Object	自动化对象引用
Method	由自动化对象执行的操作
Parameters	要传递到方法或从方法返回的信息。参数数目由所调用的方法定义。参数可以是只读或输入/输出型。

限制

调用方法受以下条件的限制：

- 方法可作为“子程序”调用（不返回值）。如果方法返回值，则将被丢弃。
- 如果要使用返回值，必须指定为未修改。方法调用不能作为表达式中的条件。
- 通常，在从执行过程中返回方法之前，脚本执行将被冻结，除非方法执行要求用户通过对话框进行交互的事务处理。
- 您必须确保传递正确的参数数目。不支持 OLE“可选”参数。

参数类型

从 InTouch 将数据类型传递到自动化对象有一些限制条件：

数据类型	可表示为
整型	常数、表达式、只读标记、读/写标记或读/写 OLE 自动化属性
实型	常数、表达式、只读标记、读/写标记或读/写 OLE 自动化属性
消息型	常数、表达式、只读标记、读/写标记或读/写 OLE 自动化属性
离散型	常数、表达式、只读标记、读/写标记或读/写 OLE 自动化属性
对象	对象引用名称或读/写 OLE 自动化属性

注意某些方法有“多种形态”，相同参数可使用多种不同的数据类型。InTouch 支持在仅输入型参数中使用此机制。

输出参数

InTouch 不能区别“仅输出”型和“输入/输出”型方法参数。如果参数可以更新，是会在其中存储值。

在以下条件下可以更新参数：

- 参数为读/写 InTouch 标记的名称。
- 参数为对象引用名称。
- 参数指定对象属性。

在下面的 InTouch 语句中，前三个参数将被更新，因为它们符合上面所列的相应选项。其它参数不会被更新。

```
%Obj1.Method(tag1, %obj2, %obj3.property1.property2, tag1 + 3,
             %obj4.property + 0);
```

返回值

为获得方法的返回值，赋值语句的左边必须为：

- 读/写 InTouch 标记名的名称。
- 对象引用名称。
- 对象属性规格。

实例 1

假定创建了一个 SerialNumbers 类的自动化对象并将其命名为 %SerialNumbers。下面的脚本语句在 %SerialNumbers 采集对象的序列号中添加序列号 “1234”。

```
returnCode = %SerialNumbers.Add("1234");
```

实例 2

假定创建了一个 Sublot 类的自动化对象并将其命名为 %Sublot_Object。您需要使用 Sublot Ship 方法，它包含自动化对象 %PrimaryAmt 和 %SecondaryAmt 的两个参数：

```
ReturnCode = %Sublot_Object.Ship (%PrimaryAmt, %SecondaryAmt,
                                  "customerName") ;
```

此处：

%PrimaryAmt	确定要交付的主数量的“数量”型对象
%SecondaryAmt	确定要交付的次数量的“数量”型自动化对象
customerName	收货客户的名称

样本

您需要向客户 (Wonderware) 交付 100 份手册（主数量），每件重达 2 磅（次数量）。在调用 Ship 方法之前，您必须创建参数自动化对象 %PrimaryAmt 和 %SecondaryAmt，并设置其属性。步骤如下所述：

1. 创建一个 Amount 对象来表示主数量，并设置属性：
```
OLE_CreateObject(%PrimaryAmt, "InTrack.Amount");
%PrimaryAmt.Quantity = 100;
%PrimaryAmt.Units = "each";
```
2. 创建一个 Amount 对象来表示次数量，并设置属性：
```
OLE_CreateObject(%SecondaryAmt, "InTrack.Amount");
%SecondaryAmt.Quantity = 2;
%SecondaryAmt.Units = "pounds";
```
3. 使用下面的语法，对 %Sublot_Object 调用 Ship 方法：
```
returnCode = %Sublot_Object.Ship(%PrimaryAmt, %SecondaryAmt,
                                 "Wonderware");
```

InTouch 错误处理

当处理错误时，确定错误是因为不正确使用 OLE 自动化而引起，还是从 OLE 服务器返回。InTouch 在消息框中识别 OLE 错误并将其记入 WWLogger。

InTouch 脚本执行不会因 OLE 错误而终止。所采取的动作取决于出错时所执行的任务：

- 如果在获取属性操作时出错，属性值将返回零或空白字符串。
- 如果在设置属性操作时出错，属性值将不会更改。
- 如果在调用方法时出错，脚本将不会调用方法，不更改任何参数值，方法的返回值为零或空白字符串。

下面五个系统报告函数有助于进行错误处理：

- OLE_GetLastObjectError
- OLE_GetLastObjectErrorMessage
- OLE_ResetObjectError
- OLE_ShowMessageOnObjectError
- OLE_IncrementOnObjectError

这些函数详述如下：

OLE_GetLastObjectError

通过 Microsoft “last error” 变量获得上次报告的错误号：

```
value = OLE_GetLastObjectError();
```

此处：

value　　存储错误号的一个整数。
错误号为内部的无符号 32 位整数。InTouch 不具有相同的整数范围，因此不能正确返回使用高指令位的错误码。

OLE_GetLastObjectErrorMessage

获取上次报告的错误消息（说明）：

```
value = OLE_GetLastObjectErrorMessage();
```

此处：

value　　设为错误消息的字符串（消息型）。
此文本也存在于错误消息框中。它包含嵌套回车和换行符。如果文本超出 InTouch 要求的 131 个字符的字符串限制，则消息会被截短。

OLE_ResetObjectError

仅当出现错误时，“last error” 值才会更改。在 OLE 自动化动作成功完成时，此函数不会重置。在某些情况下，通过将 “last error” 设为已知状态、执行某些 OLE 自动化操作、然后测试 “last error” 值的变化，可以较容易地进行应用程序开发。此函数会将 Microsoft 的内部 “last error” 值设为零。此函数的语法如下：

```
OLE_ResetObjectError();
```

OLE_ShowMessageOnObjectError

当出现 OLE 接口错误或 OLE 异常时，此函数控制显示和要求操作员输入。当显示此消息框时，导致错误出现的脚本将被冻结运行，直到消息框被确认为止（例如，单击“确定”按钮）。后台脚本将继续执行。

OLE 接口错误包括无法与 Microsoft Windows-NT 操作系统所提供的 Microsoft OLE 结构通讯。它不会反映由 OLE 服务器所识别的处理错误。

OLE 异常可能由 OLE 结构或 OLE 服务器引起。

此函数的语法如下：

```
OLE_ShowMessageOnObjectError(discreteTag);
```

此处：

discreteTag	定义布尔（离散）值的名称，指示在出现 OLE 错误或异常时，是否向用户显示消息框，此处： -1 = 显示消息框（缺省） 0 = 不显示消息框

下面的脚本禁止在使用 Create 方法调用对象 %Lot 时显示 OLE 错误消息框。在调用方法后，OLE 错误消息框重新启用，以显示任何后续错误。

```
OLE_ShowMessageOnObjectError(0);
%Lot.Create("abc");
OLE_ShowMessageOnObjectError(1);
```

OLE_IncrementOnObjectError

此函数在出现 OLE 接口错误或 OLE 异常的情况下递增整型标记名。

注意： 如果在启用离散标记的情况下执行 OLE_ShowMessageOnObjectError 函数，此函数会在显示并确认 OLE 错误消息框*之后*递增值。

OLE 接口错误包括无法与 Microsoft Windows-NT 操作系统所提供的 Microsoft OLE 结构通讯。它不会反映由 OLE 服务器所识别的处理错误。

OLE 异常可能由 OLE 结构或 OLE 服务器引起。

此函数所指定的标记会在在每次有错误出现时递增，直至另一调用指定了不同的标记。在执行 WindowViewer 时，不会记忆标记。

此函数的语法如下：

```
OLE_IncrementOnObjectError(integerTag);
```

此处：

integerTag　定义出现错误或异常时要递增的整型标记名。

下面的脚本在每次出现 OLE 接口错误或 OLE 异常时，将 errorCount 标记所引用的变量值递增 1。

```
OLE_IncrementOnObjectError(errorCount);
```

常见自动化对象错误

使用 InTouch OLE 自动化扩展时的常见错误及其解决方法列出如下：

- **OLE 对象引用名称没有定义。**

  对象引用名称在与对象关联之前，已用于引用属性 (%object.property) 或调用方法 (%object.method(parameter))。检查对象引用名称是否拼写错误。通过使用 InTouch 系统函数 OLE_CreateObject 或对另一对象指定对象引用，检查对象的关联名称。

- **未知的属性或方法名称。**

  检查属性或方法名称是否正确拼写。

- **企图修改只读属性。**

  当只读属性位于赋值语句的左边时，此错误很明显。如果作为方法调用的返回参数更新，则会很难查出。

- **输入不匹配。**

  作为参数传递以进行更新的数据不正确。这表示自动数据转换没有足够的信息来采取正确的措施。对 InTouch 标记手动指定正确类型的数据，并使用标记。

- **无效的类字符串。**

  这通常表示 OLE_CreateObject 函数调用中的类名（第二个参数）拼写错误。

- **参数不可选。**

  检查所传递的参数数目是否正确。

- **意外结果但未报告错误。**

  在运行时执行数据类型转换。如不小心对整型值赋予字符串，并不会产生错误。但是，如果字符串不是以数字开头，则您可能获得零结果。

- **对象引用路径元素不是 OLE 对象。**

  如果路径中的某个属性不是嵌套对象或对象引用，则当使用多个“点”操作符时会出现此错误。

附 录 A

QuickScript 函数错误排解

某些 InTouch QuickScript 函数会根据执行状态来返回值。此值称为结果码或错误码，可用于决定函数执行的相对成功性。本附录介绍窗口控件、配方和 SQL QuickScript 函数的结果码，此外还向您介绍用于获取数据集信息和控制图操作的 SPC DDE 项目。

窗口控件和分布式报警的错误消息

窗口控件和分布式报警 QuickScript 函数会根据 QuickScript 函数的处理结果来返回值。用于错误诊断的返回值可赋给整型标记名。例如：

```
ErrorNumber = wcGetItem("ControlName", Number, Tagname);
```

此处：

ErrorNumber 是保存返回的错误值的整型标记名。函数的返回值可以传递给 **wcErrorMessage()** 函数，**wcErrorMessage()** 函数返回有关错误的字符串说明。例如：

```
ErrorMsg = wcErrorMessge(ErrorNumber);
```

此处：

ErrorMsg 是保存返回错误的文本说明的消息型标记名。

下表介绍了 ***ErrorNumber*** 数字值：

错误号	描述
0	成功
-1	一般错误
-2	可用内存不足
-3	属性为只读
-4	指定项目已经存在
-5	未知的对象名
-6	未知的属性名
-*x**	未知的错误

* -*x* 表示任何其它数字。

配方脚本函数错误排解

要检索配方函数的错误码， 它必须等于一个 InTouch 整型标记名。例如：

```
ErrorCode = RecipeLoad(FileName, UnitName, RecipeName);
```

如果成功，**RecipeLoad()** 函数将标记名 **ErrorCode** 的值设为“0”；如果 **RecipeLoad()** 失败，它会将模拟型标记名 **ErrorCode** 设为表示特定错误条件的编号。以下列出可能的错误码以及它们的相应错误消息和描述：

值	错误消息	描述
0	成功	所调用的配方函数成功执行。
-1	无这种配方模板	指定的配方模板文件名不存在。
-2	View 没有激活	由于 WindowViewer 没有运行，另一程序所调用的配方函数不能执行。
-3	内存不够	没有足够内存来完成当前任务。
-4	在配方模板文件中行太长	配方模板文件中的某一行超出最大允许长度。
-5	配方文件内的行被截断	配方模板文件中的某一行被截断。
-6	不是一个有效的配方文件	指定的文件名不是有效的.csv 配方模板文件。 有关单元或配方的详细信息，请参阅联机《*配方管理器用户指南*》。
-7	需要“单元”或“配方”	配方模板文件缺少单元名或配方名。 有关单元或配方的详细信息，请参阅联机《*配方管理器用户指南*》。
-8	在配方模板文件中无定义单元	配方模板文件“单元定义”模板中没有定义单元。
-9	在配方模板文件中配方名未发现	配方模板文件中未定义指定的配方名。
-10	在配方模板文件中单元名没找到	单元定义模板文件中未定义指定的单元名。
-12	需要“模拟”、“离散”、“消息”	配方模板文件中输入的项目类型不正确。有效类型只限于“模拟”、“离散”和“消息”。

值	错误消息	描述
-13	标记名类型与“模拟”、“离散”或“消息”不匹配	指定标记名的项目类型不正确，例如，配方项目定义为“模拟”，但在单元中却为其定义了消息型标记名。
-14	无效的离散值，需要“0”，“1”	配方模板文件中的离散量输入了不正确的值。离散量的有效值为 0 或 1。
-15	无法打开临时文件	无法打开临时文件，磁盘空间可能不够。
-16	在保存配方模板文件时写错误	保存配方模板文件时出现写错误。
-17	没有选定用户	用户在“选择配方”对话框中选择了“取消”而不是配方名。
-19	其它应用程序正在使用配方模板	指定的配方模板文件已被打开，所以 WindowViewer 不能访问它。

显示配方错误码消息

每个配方函数都会返回一个数字，表示函数的错误条件。通过在 InTouch 数据改变 QuickScript 中使用 **RecipeGetMessage()** 函数，相应的错误代码可以写入模拟型标记名，而关联的错误代码消息可以写入消息型标记名中：

为实现这一点，使用下面的数据改变 QuickScript：

RecipeGetMessage(ErrorCode, ErrorMessage, 131);

每次模拟标记名 **ErrorCode** 的值改变时，将自动执行此 QuickScript。当执行此 QuickScript 时，**RecipeGetMessage()** 函数会读取标记名 **ErrorCode** 的当前数值，并将与该数值关联的消息返回给标记名 **ErrorMessage**。

SPC DDE 项名

SPC DDE 项可用于获取数据集信息和控制图表操作。*应用程序名*是 SPC。*主题名*是数据集名。

注意：除了分布式 SPC *主题名*必须为间接数据集名称之外，所有 SPC DDE 项可象用于非分布式 SPC 一样用于分布式 SPC。

SPC 控制和显示 DDE 项

“控制”和“显示”DDE 项用于控制和显示有关主题数据集的信息。“控制”DDE 项由所有节点共享。它们是远程数据集的采集产品的数据集值。“显示”DDE 项对于每个节点是本地的。它们是本地节点上所显示产品的样本值，

样本修改可应用于本地任何数据集的采集和显示产品。通过单击图表显示所做的修改将影响显示产品。SPC DDE 项可修改所采集的产品。显示的产品和采集的产品有它们自己的当前样本，是最近记录的样本。

程序将所为采集和显示的产品评价并保存报警。它们仅在运行时针对所采集的产品进行报告。

注意：在加入显示和采集产品后，许多 SPC DDE 项仅应用于采集产品。这些项目在下表中以 SPC DDE 项名前加一个星号 (*) 表示。

项目名	DDE 类型	访问	描述
AutoCollection	离散型	读/写	启用/禁用自动数据采集。
***CalculateControlLimits**	离散型	读/写	设为 1 以启动控制限计算。
DatasetName	消息型 (32)	读/写	设置由间接数据集使用的数据集名。
HistogramLCL	实型	只读	根据密度显示直方图的下控制限。
HistogramUCL	实型	只读	根据密度显示直方图的上控制限。
Kurtosis	实型	只读	直方图的分布形状。
LastSampleDisplayed	整型	读/写	设置由数据集显示的最后一个样本号。

项目名	DDE 类型	访问	描述
***ManualInputDialog**	离散型	读/写	设置 1 以显示内置的“手工输入”对话框。
MeasurementsPerSample	整型	只读	显示每个样本所配置的测量次数。
NewProduct	消息型 (32)	读/写	用来创建新产品名。
NewProductCtrlTitle	消息型 (32)	读/写	用来设置由 NewProduct 函数建立的新产品的控制图表标题。
NewProductParetoTitle	消息型 (32)	读/写	用来设置由 NewProduct 函数建立的新产品的巴累托图表标题。
NewProductHistTitle	消息型 (32)	读/写	用来设置由 NewProduct 函数建立的新产品的直方图标题。
***ProductCollected**	消息型 (32)	读/写	更改由数据集采集的产品名。
ProductDisplayed	消息型 (32)	读/写	更改由数据集显示的产品名。
SampleSize	整型	只读	NP 数据集的样本大小。
SamplesPerControlChart	整型	读/写	设置控制图中显示的样本数。
SamplesPerHistogram	整型	读/写	设置直方图中显示的样本数。
SamplesPerLimitCalc	整型	读/写	设置控制限计算中使用的样本数。
SamplesPerPareto	整型	读/写	设置巴累托图显示中使用的样本数。
SelSPCOutSpecMsg	消息型	只读	“超出规格限的样本”的报警消息标记。
Skewness	实型	只读	显示从直方图平均值中得出的偏差。
SPCAllowSampDelMod	离散型	读/写	切换打开或关闭右击菜单选项：“删除样本”和“修改样本”。

项目名	DDE 类型	访问	描述
SPCConnection	离散型	只读	设置与服务器的连接丢失，则设为零。
SPCConnectType	消息型	只读	显示是否节点是作为代理还是客户端连接。
SPCLowDBSpace	离散型	只读	用来监视 Microsoft SQL Server 数据库。只与 Microsoft SQL Server 数据库同时工作。当数据库空间小时，其值等于 1。可用来停止 AutoCollection，并提醒操作员分配更多空间或更多释放硬盘空间。它会依据 SQL 数据库状态在 1 和零之间自动切换。
SPCResetRunRules	离散型	读/写	用于重置对所采样本所应用的运行规则。此项目很重要，因为在为新批产品采集数据时，您可能不想将运行规则应用于新批样本。您可能需要重置新批规则。

SPC 当前样本 DDE 项

所有“当前样本”DDE 项与给定数据集上次采集的样品有关。它们可用于改变与数据集名称关联的原始数据和极限。要更改有关当前样本的信息，您必须写入适当的 DDE 项，然后将 **CurrentUpdate** DDE 项设为 1。这相当于重新输入样本并执行任何所需的计算。SPC 程序会在输入样本后将 **CurrentUpdate** DDE 项重置为 **0**。一旦下一个要采集的样本启动采集周期，当前样本 DDE 项将不能再更新。

当前样本 DDE 项在所有节点中共享。这些项值代表采集产品的上次采集样品。

对于分布式 SPC，最初所有值均设为 0。SPC 连接到数据集并且每 5 秒检查新数据一次。当发现新数据时项目值将被更新。对当前样本值的修改将存储到本地缓冲区中，直至 **CurrentUpdate** 项目设为 1，随后这些值将放入当前样本包，并发送至远程数据集节点供分析和存储。当前针对不同采集产品进行的样品修改，以及不是最新记录样品的当前样本号会被服务器拒绝。

注意：在加入显示和采集产品后，所有“当前”SPC DDE 项将只应用于采集产品。

项目名	DDE 类型	访问	描述
CurrentCauseCode	整型	读/写	设置当前样本的特殊原因代码。
CurrentCauseString	消息型 (128)	只读	为当前样本显示特殊原因代码的描述。
CurrentComment	消息型 (50)	读/写	用来读/写与当前样本有关的任何其它注释。
CurrentCp	实型	只读	显示当前样本容量。
CurrentCpk	实型	只读	显示当前样本的中心容量。
CurrentDate	消息型 (8)	读/写	以格式 **DD/MM/YY** 来设置当前样本的日期。如果输入错误，缺省为当前日期。
CurrentFlag	离散型	读/写	设置当前样本的标帜。

项目名	DDE 类型	访问	描述
CurrentIgnoreValue	离散型	读/写	当控制图为自动调节比例时，设置要忽略的当前样本。
CurrentM*x*	实型	读/写	设置当前样本的个别测量值（*x*=1 到 25）。
CurrentR	实型	只读	显示当前样本的范围。
CurrentRBar	实型	读/写	设置当前样本的平均范围。
CurrentRLCL	实型	读/写	设置范围的下控限。
CurrentRUCL	实型	读/写	设置范围的上控限。
CurrentSample	实型	只读	显示最后样本点的值（即 X、C、P）。
CurrentSampleBar	实型	读/写	设置此样本点的当前样本平均值。
CurrentSampleNumber	整型	只读	显示采集的最后样本号。
CurrentTarget	实型	读/写	设置此样本点的目标值。
CurrentTime	消息型 (8)	读/写	为以 **HH:MM:SS** 为格式的当前样本设置时间。如果输入错误，缺省为当前时间。
CurrentUpdate	离散型	读/写	要更改在任意当前字段中输入的样本信息，可将此项设为 1。
CurrentXLCL	实型	读/写	设置当前样本的下控限 (LCL)。
CurrentXLSL	实型	读/写	设置当前样本的下规格限 (LSL)。
CurrentXUCL	实型	读/写	设置当前样本的上控限 (UCL)。
CurrentXUSL	实型	读/写	设置当前样本的上规格限 (USL)

项目名	DDE 类型	访问	描述
SPC2L3Out2SD	整型	只读	报警“最后三个样本中的两个样本超过两个标准偏差（同侧）”的报警计数器。
SPC2L3Out2SDMsg	消息型	只读	“最后三个样本中的两个样本超过两个标准偏差（同侧）”的报警消息标记。
SPC4L5Out1SD	整型	只读	报警“最后五个样本中的四个样本超过一个标准偏差（同侧）”的报警计数器。
SPC4L5Out1SDMsg	消息型	只读	“最后五个样本中的四个样本超过一个标准偏差（同侧）”的报警消息标记。
SPCConSampAltUpDn	整型	只读	报警“上下交替的连续样本”的报警计数器。
SPCConSampAltUpDnMsg	消息型	只读	“上下交替的连续样本”的报警消息标记。
SPCConSampln1SD	整型	只读	报警“1 标准偏差内的连续样本”的报警计数器。
SPCConSampln1SDMsg	消息型	只读	“1 标准偏差内的连续样本”的报警消息标记。
SPCConSamplncDec	整型	只读	报警“递增或递减的连续样本”的报警计数器。
SPCConSamplncDecMsg	消息型	只读	“递增或递减的连续样本”的报警消息标记。
SPCConSampOneSideCL	整型	只读	报警“连续样本在中心线的一侧”的报警计数器。
SPCConSampOneSideCLMsg	消息型	只读	“连续样本在中线的一侧”的报警消息标记。
SPCConSampOut1SD	整型	只读	报警“连续样本超过标准偏差 1”的报警计数器。
SPCConSampOut1SDMsg	消息型	只读	“连续样本超过标准偏差 1”的报警消息标记。
SPCNLNOutNSD	整型	只读	报警“最新？样本的？超出 ？ sd 的范围”的报警标记。
SPCNLNOutNSDMsg	消息型	只读	“最新？样本的？超出 ？ sd 的范围”的报警标记。
SPCNLNOutNSDSS	整型	只读	报警“最新？样本的？超出 ？ sd 的范围(同一侧) ”的报警标记。

项目名	DDE 类型	访问	描述
SPCNLNOutNSDSSMsg	消息型	只读	“最新？样本的？超出 ？sd 的范围 (同一侧) ”的报警标记。
SPCOutRCtrl	整型	只读	范围图表报警“超出控制限制的范围”的报警计数器。
SPCOutRCtrlMsg	消息型	只读	范围图表报警“超出控制限制的范围”的报警消息。
SPCOutXCtrl	整型	只读	X 图表报警“超出控制限制的样本”的报警计数器。
SPCOutXCtrlMsg	消息型	只读	图表报警“超出控制限制的样本”的报警消息。
SPCOutSpec	整型	只读	报警“超出规格限制的样本”的报警计数器。
SPCOutSpecMsg	消息型	只读	“超出规格限的样本”的报警消息标记。
SPCRecalculateCp	离散型	读/写	当设为 1 时，引擎会在采集下一样本时，重新计算当前数据集的 Cp 和 Cpk 值。在采集该样本后，本项目值将重置为 "0"。要重新计算 Cp 和 Cpk，请再次将此值设为 "1"。当设为 1 时，下次采集的样本将重新计算 Cp 和 Cpk 值。在采集样本后，再自动重置为 0。 **注意：**此项目仅影响 Cp 和 Cpk 值，它不会影响控制限或运行规则。
SPCResetAlarmCounters	离散型	读/写	重置所有报警计数器。
SPCResetRunRules	离散型	读/写	用于重置对所采样本所应用的运行规则。应用于当前采集的产品。当开启时，运行规则将重置，先前的样本将不会应用于诸如“五个样本中有四个样本超过 1 个标准偏差“之类的报警计算中。此操作只进行一次，运行规则将恢复正常。要再次执行重置操作，必须重置本项目，然后再次开启。

SPC 手工输入 DDE 项

“手工输入”DDE 项用于创建自定义手动输入窗口。要使用手工输入项，请设置适当的项目值然后将“**MI_Save**”DDE 项设为 1。这将导致其它 MI 域的信息作为新样本输入。在样本输入后，SPC 程序将置 **MI_Save** DDE 项为（零）。

对于分布式 SPC，手工输入 DDE 项在每个节点上是彼此独立的。这些值将存储到每个节点的本地缓冲区直到 DDE 项“**MI_Save**”设为 1。一旦“**MI_Save**”设为 1，这些值将放入手工输入包并发送至远程数据集节点供分析和存储。

注意：在加入显示和采集产品后，所有“当前”SPC DDE 项将只应用于采集产品。

项目名	DDE 类型	访问	描述
MI_CauseCode	整型	读/写	为手工输入样本设置特殊原因代码。
MI_CauseString	消息型 (127)	只读	为样本显示特殊应用程序号码输入的描述。
MI_Comment	消息型 (50)	读/写	用于读/写为样本输入的任何其它方面的注释。
MI_Date	消息型 (8)	读/写	设置当前样本的日期。日期必须以格式 **DD/MM/YY** 输入。如果输入不正确，日期将缺省为当前日期。
MI_Flag	离散型	读/写	设置手工输入样本的标帜。
MI_IgnoreValue	离散型	读/写	当控制图为自动调节比例时，设置要忽略的当前样本。
MI_M*x*	实型	读/写	为指定的手工输入测量单位（*x*=1 到 25）设值。
MI_Save	离散型	读/写	将其它 MI 字段中手工输入的信息保存为新样本。 **注意：**当 **MI_Save** 项设为 1 时，所有 **MI** 项的值将写入各自的当前 DDE 项，并且 **CurrentSampleNumber** 项将编号为 1。
MI_Time	消息型 (8)	读/写	设置当前样本的时间。时间必须以格式 **HH:MM:SS** 输入。如果输入不正确，时间缺省为当前时间。

SPC 选择 DDE 项

“选择”DDE 项可用于查看有关任何样本的详细信息。DDE 项“选择”用于输入要显示的样本号。一旦键入，SPC 程序将用“选择”样本号的详细信息来更新所有其它“选择”项。

不能更改旧的数据，但通过设置适当的项目然后将“**SelectionUpdate**”项设为 1，可添加特殊原因代码、标帜和（或）注释。

这使得可以用新值来修改所选样本记录。SPC 程序会在输入更新的样本后将 **SelectionUpdate** DDE 项重置为 **0**。

对于分布式 SPC，选定的样本 DDE 项在每个节点上是彼此独立的。它们是远程节点为采集产品的指定样本号所记录的样本值。当“选择”DDE 项设为样本号后，就可以从远程节点样本文件中获取样本信息。不能修改旧数据，但通过设置适当的项目然后将 **SeletionUpdate** 项设为 1，可添加特殊原因代码、标帜和注释。当 **SelectionUpdate** 设为 1 时，特殊原因代码、注释、标帜和忽略值项将通过数据包发送至远程节点以供存储。

注意：在加入显示和采集产品后，所有“选择”SPC DDE 项将只应用于采集产品。

项目名	DDE 类型	访问	描述
选择	整型	读/写	将此项设置为样本号将可以用该样本的数据适当更新所有选择项目。
SelectionCauseCode	整型	读/写	为选定的样本设置特殊原因代码。
SelectionCauseString	消息型 (128)	只读	显示输入的特殊原因程序的描述。
SelectionComment	消息型 (50)	读/写	用于读/写为所选样本输入的任何其它注释。
SelectionCp	实型	只读	显示所选样本的容量。
SelectionCpk	实型	只读	显示所选样本的中心容量。
SelectionDate	消息型 (8)	只读	显示所选样本的日期。
SelectionFlag	离散型	读/写	设置所选样本的标帜。
SelectionIgnoreValue	离散型	读/写	当控制图表为自动调节比例时，设置要忽略的选定样本。

项目名	DDE 类型	访问	描述
SelectionMx	实型	只读	显示组成样本的个别测量 (*x*=1-25) 值。
SelectionProduct	消息型 (32)	只读	显示所选样本的产品名。
SelectionRUCL	实型	只读	为所选样本显示范围 **UCL**。
SelectionRLCL	实型	只读	为所选样本显示范围 **LCL**。
SelectionR	实型	只读	为所选样本显示**范围**。
SelectionRBAR	实型	只读	为所选样本显示**平均**范围。
SelectionSample	实型	只读	显示所选样本点的值。
SelectionSampleBar	实型	只读	在所选样本点显示所选样本的平均值。
SelectionTarget	实型	只读	在所选样本显示**目标**值。
SelectionTime	消息型 (8)	只读	在所选样本显示**时间**值。
SelectionUpdate	离散型	读/写	更新“选择”字段中的更改。
SelectionXUSL	实型	只读	显示样本的上规格限。
SelectionXLSL	实型	只读	显示样本的下规格限。
SelectionXUCL	实型	只读	显示样本的上控限。
SelectionXLCL	实型	只读	显示样本的下控限。
SelSPC2L3Out2SDMsg	消息型	只读	“最后三个样本中的两个样本超过 2 个标准偏差（同侧）”的报警消息标记。
SelSPC4L5Out1SDMsg	消息型	只读	“最后五个样本中的四个样本超过一个标准偏差（同侧）”的报警消息标记。
SelSPCConSampAltUpDnMsg	整型	只读	警报“上下交替的连续样本 ”的警报消息。
SelSPCConSampln1SDMsg	消息型	只读	“1 标准偏差内的连续样本”的报警消息标记。
SelSPCConSamplncDecMsg	消息型	只读	“递增或递减的连续样本”的报警消息标记。
SelSPCConSampOneSideCLMsg	消息型	只读	“连续样本在中线的一侧”的报警消息标记。
SelSPCConSampOut1SDMsg	消息型	只读	“连续样本超过标准偏差 1”的报警消息标记。

项目名	DDE 类型	访问	描述
SelSPCNLNOutNSDMsg	消息型	只读	“最新？样本的？超出 ？sd 的范围”的报警标记。
SelSPCNLNOutNSDSSMsg	消息型	只读	“最新？样本的？超出 ？sd 的范围 (同一侧) ”的报警标记。
SelSPCOutRCtrlMsg	消息型	只读	范围图表报警“超出控制限制的范围”的报警消息。
SelSPCOutXCtrlMsg	消息型	只读	图表报警“超出控制限制的样本”的报警消息。
SelSPCOutSpecMsg	消息型	只读	“超出规格限的样本”的报警消息标记。

注意：可设置多个间接数据集并链接至同一个实数据集。每个间接数据集“选择”的值可设为一个不同的样本号。这使您可以在一个数据集里查看多个样本的信息。

SQL 脚本函数错误排解

所有 SQL 函数均会返回一个可用于除错的 *ResultCode*。**SQLErrorMsg()** 函数会返回与 *ResultCode* 关联的错误消息。例如：

```
ErrorMsg=SQLErrorMsg(ResultCode);
```

此处：

ErrorMsg 是内存消息型标记名。
ResultCode 是从先前 SQL 函数获得的整型值。

结果码错误消息

对于此处没有记录的结果码，请参考您的特定数据库手册并确保检查 Wonderware Logger，以获得任何附加信息。

SQLErrorMsg() 函数将设置 InTouch 消息标记 *ErrorMsg* 的值。下面列出可能出现的一些 SQL 结果码及其相应的错误消息和描述：

结果码	错误消息	描述
0	**没有发生错误。**	命令成功执行
-5	**无更多的行可取用**	已到达表格中的最后一条记录
-1001	**内存不够**	没有足够的内存来执行此函数
-1002	**无效连接**	传递给函数的 ConnectionId 无效
-1003	**未找到绑定表**	指定的绑定表名称不存在
-1004	**找不到模板**	指定的表格模板不存在。
-1005	**内部错误**	出现内部出错，. 请致电“技术支持”。
-1006	**字符串为空**	警告 – 从数据库读取的字符串是空的。
-1007	**字符串被截短**	警告 – 从数据库中读取的字符串超过 131 个字符，所选字符串将被截短。
-1008	**无 Where 子句**	Delete 上没有 Where 子句。

结果码	错误消息	描述
-1009	**连接失败**	欲知关于连接失败的更多详细描述，参见 Wonderware 日志。
-1010	**连接字符串的“DB=部分”上指定的数据库不存在**	指定的数据库不存在。
-1011	**没有选择行**	未先执行 **SQLSelect()** 命令前，而试图调用 **SQLNumRows()**、**SQLFirst()**、**SQLNext()** 或 **SQLPrev()**。
-4149	**所给连接、语句或查询句柄无效**	列类型可能定义错误。假如，如果表格模板定义为字符列类型，而不是 dBASE 文件字符，则会返回错误。

特定数据库错误消息

Oracle

错误消息	答案
ORA-03112 – 主机字符串语法错误	如果您没有运行 **NETINIT.EXE**，请将它置入 Windows Startup 组中。如果您正在运行 **NETINIT.EXE**，并希望建立一个以上的连接或会话，则需要分配内存。为此，将 **CONFIG.ORA** 文件中的 **WIN_REMOTE_SESSIONS** 参数值设置为所需的连接数。下面的例子分配 4 个连接的内存： **WIN_REMOTE_SESSIONS=4**
ORA-3121 没有连接接口驱动程序	在进入 Windows 并使用 InTouch SQL Access 之前，为网络系统适当启动的 SQL*NET TSR。
ORA-6435 - NetBIOS:不能添加本地名称到名称表格	必须运行网络软件（Novell、LAN Manager 等）
ORA-09301 -本地核心只在标准模式中被支持	在连接字符串中指定服务器名 (**"SRVR="**)
ORA-06430 -不能连接	确认连接字符串中的属性准确无误，并且次序正确。

Sybase 或 Microsoft SQL Server

错误消息	答案
您不能一次使用多条语句	在执行 **SQLSelect()** 后您正试图执行一个 SQL 命令。执行 **SQLEnd()** 来从 **SQLSelect()** 释放系统资源或，或在第二个语句使用不同的 ConnectionId。
处理命令的内存不够	尝试重新启动客户端工作站。
无效对象名称，表名称	表名称不存在于您当前使用的数据库中。尝试 DB=database name。

dBASE

错误消息	答案
文件或 DLL 没有发现	对于 Windows，QEDBF.DLL 必须在当前目录或在 DOS 路径下的窗口系统目录。
无效连接	确定您的路径中具有适当的 DLL。若要 DBF 支持，您将需要 QLDBF.DLL。
连接或所给语句句柄无效	有关详细消息，请查阅 Wonderware Logger，从中可能发现 SQL 语句中的更多语法错误。

索引

符号

D

G

I

O

Q

R

S

W

四划

六划

七划

八划

九划

十划

十一划

十二划

十三划